AD-A062 389 NOTRE DAME UNIV IND DEPT OF AEROSPACE AND MECHANICA--ETC F/G 12/2
OPTIMIZATION OF MULTIPLE-RESPONSE SIMULATION MODELS.(U)
OCT 78 W E BILES N00014-76-C-1021

UNCLASSIFIED NL

1 OF 2
ADA
062389

OF 2

DA

2389

LEVEL II

12

AD A062389

DDC FILE COPY

DDC
RECEIVED
DEC 14 1978
F

INDUSTRIAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA 46556

Department of Aerospace and Mechanical Engineering

78 12 11 160

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AD A062389

DDC FILE COPY

DDC
DEC 14 1978
F

9 FINAL REPORT.
1 Sep 76-31 Oct 78

6 OPTIMIZATION OF MULTIPLE-RESPONSE SIMULATION MODELS.

10 William E./Biles
Department of Aerospace and Mechanical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

This document has been approved for public release and sale; its distribution is unlimited.

11 31 Oct 78

12 96p.

October 31, 1978

This research was sponsored by
Naval Analysis Programs
Office of Naval Research
Contract N00014-76-C-1021
15 NR 277-237

16 RRØ1411
17 RRØ141101

Reproduction in whole or in part is permitted for any purpose of the United States Government. Available for public release; distribution unlimited.

401 829

JOB

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OPTIMIZATION OF MULTIPLE RESPONSE SIMULATION MODELS.		5. TYPE OF REPORT & PERIOD COVERED Final Report September 1,1976-Oct. 31,1978
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. William E. Biles		8. CONTRACT OR GRANT NUMBER(s) N0014-76-C-1021
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Aerospace and Mechanical Engr. University of Notre Dame Notre Dame, IN 46556		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153 N RR014-11-01 NR 277-237
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Analysis Programs (Code 431) Office of Naval Research Arlington, VA 22217		12. REPORT DATE October 31, 1978
		13. NUMBER OF PAGES 91
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) N/A		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Available for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Available for public release; distribution unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Optimization
Simulation
Response Surface Methodology

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes several computerized multiple-variable, multiple-response optimization procedures for use in connection with computer simulation models. These procedures include (1) "complex" search, (2) a sequential first-order response surface approach, (3) a second-order response surface method, and (4) "complex" search followed by second-order response surface analysis. These four optimization procedures are evaluated with (1) a stochastic inventory simulation model, (2) a tank duel simulation, and (3) a naval minefield evaluation model.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

78 12 11 160

TABLE OF CONTENTS

ACCESSION for
NTIS
DDC
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
SPECIAL
A 23 CP

ABSTRACT

This report describes several computerized multiple-variable, multiple-response optimization procedures developed for use in connection with simulation models. The four optimization procedures evaluated in this research were as follows:

1. Box's complex search method [8];
2. A sequential first-order response surface approach [2,3];
3. A second-order response-surface experimental design followed by mathematical optimization;
4. A Box complex search followed by a second-order response surface optimization (methods 1 and 3 combined).

Each of these optimization methods involved an adaptation of the original procedure on which it was based, in order to accommodate multiple simulation responses η_j, $j=1, \ldots, m$.

These four optimization procedures were evaluated with three computer simulation models:

i. A stochastic inventory model [15,16];
ii. A tank duel model [21];
iii. A minefield evaluation model [1].

Exploratory work was also performed with a simulation model of a missile attack on a surface fleet [18].

This research has produced workable optimization procedures for interfacing with simulation models of naval operations, but it is concluded that this interface must be managed on a "customized" basis for each simulation model due to the variety of approaches employed in naval systems modeling.

INTRODUCTION

The design and analysis of complex systems, especially those involving probabilistic or stochastic elements, often necessitates the use of digital computer simulation. A problem-solving procedure for defining and analyzing a model of a system, simulation affords the analyst the opportunity to evaluate complex systems, (a) without constructing them if they are proposed systems, (b) without interrupting their operation if they are actual systems, or (c) without damaging or destroying them if the purpose of simulation is to test the effectiveness of the system in a hazardous operating environment.

Naval operations present especially challenging problems for the simulationist. The purpose in simulating naval operations would very likely be to evaluate the effectiveness of certain offensive or defensive fleet configurations and tactics. Large numbers of controllable input variables, uncontrollable input variables, and measures of system effectiveness are often involved in realistic models of naval operations. This not only complicates model development, but hampers experimentation with the model as well.

Simulation can be defined as the establishment of a mathematical-logical model of a system and the experimental manipulation of that model on a digital computer. This paper concentrates on the latter aspect, particularly upon means of determining an "optimal" design through experimentation with a digital computer simulation model. (The word "optimal" is used guardedly here, since the randomness inherent in computer simulation contradicts the classical notion of an optimal solution.) This research has explored procedures for combining computer simulation, response surface methodology and mathematical programming to seek an estimated "optimal"

solution. These techniques are demonstrated and compared using simulation models of (1) a stochastic inventory system (2) a task duel, and (3) minefield evaluation.

BACKGROUND

A computer simulation model can be regarded as a "black-box", as illustrated in Figure 1, in which values for n controllable input variables x_i, $i=1,\ldots,n$ are combined in some manner to produce values for a set of m output or response variables $\eta_j, j=1,\ldots,m$. These responses are measures of merit or effectiveness for the system being modeled. It is typically true that the system responses are affected by a set of p uncontrollable factors, z_k, $k=1,\ldots,p$.

The execution of the computer simulation model, either for some fixed simulated time t or until r_j realizations of the j-th system response have been obtained, produces m simulated responses.

$$y_j = g_j\,(x_1,\ldots,x_n \mid z_1,\ldots,z_p) + \varepsilon_j,\ j=1,\ldots,m \tag{1}$$

The quantity y_j is an estimate of the true system response η_j, and is found from the relation

$$y_j = \frac{1}{r_j}\sum_{\ell=1}^{r_j} \xi_{j\ell} \quad ,\ j=1,\ldots,m \tag{2}$$

where $\xi_{j\ell}$ is the individual value of the j-th system response at the ℓ-th realization and r_j is the number of realizations of this response. Equation (2) suggests that, for a given execution, or simulation trial, the r_j, $j=1,\ldots,m$ need not be the same. The simulation trial also produces an unbiased estimate of the variance of the j-th system response; that is,

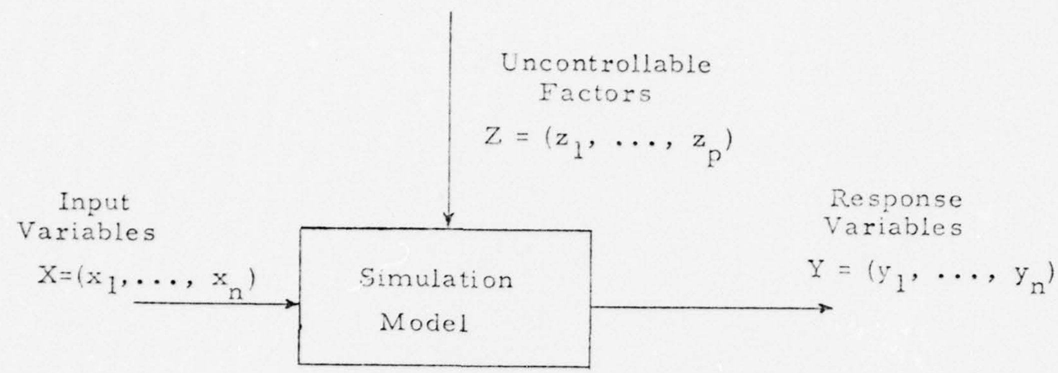


Figure 1. A "black-box" view of computer simulation

$$s_j^2 = \frac{1}{(r_j - 1)} \sum_{\ell=1}^{r_j} (\xi_{j\ell} - y_j)^2 \tag{3}$$

or more conveniently

$$s_j^2 = \frac{1}{(r_j-1)} \left[\sum_{\ell=1}^{r_j} \xi_{j\ell}^2 - r_j y_j^2 \right] \tag{3a}$$

Fishman [12] and Kleijnen [19] describe other procedures for computing s_j^2 in the case of serial correlation of the relationships $\xi_{j\ell}$.

The structure of equation (1) must be carefully examined in order to develop the basic framework for the procedures presented here. It is stated that y_j is an estimate of the true response η_j ; hence,

$$y_j = \eta_j + \varepsilon_j \tag{4}$$

The ε_j term reflects the random variation inherent in the probabilistic system under study, and can be regarded as sampling error. The uncontrollable factors, the z_k, $k=1,\ldots,p$ terms in equation (1), can be viewed as contributing to the random variation ε_j. Thus equation (1) can be simplified to

$$y_j = g_j(x_1,\ldots,x_n) + \varepsilon_j, \qquad j=1,\ldots,m \tag{5}$$

The estimate y_j is therefore a random variable with mean

$$E(y_j) = \eta_j = g_j(x_1,\ldots,x_n), \quad j=1,\ldots,m \tag{6}$$

and variance

$$\mathrm{Var}(y_j) = \mathrm{Var}(\varepsilon_j), \qquad j=1,\ldots,m \tag{7}$$

This statement suggests that the true response η_j is a point on an unknown $(n+1)$-dimensional hypersurface $g_j(X)$ at a given point $X = (x_1,\ldots,x_n)$ and that the simulation-generated response y_j is a variate from a probability

distribution about the true value η_j at this point X.

The response surface methods employed in this methodology require the assumption and the random error ε_j be normally distributed with mean zero and variance σ_j^2; that is,

$$E(y_j | x_1, \ldots, x_n) = \eta_j \qquad j=1,\ldots,m \tag{8}$$

$$\mathrm{Var}(y_j | x_1, \ldots, x_n) = \sigma_j^2, \quad j=1,\ldots,m \tag{9}$$

It is also required that $\mathrm{Var}(y_j | x_1, \ldots, x_n)$ be the same at any point on the (n+1)-dimensional hypersurface.

Note that there are m distinct (n+1)-dimensional hypersurfaces, each having its own characteristic random variation. The problem at hand is to have the computer simulation model which produces these m responses be either automatically or interactively controlled by an "optimization" procedure, as illustrated in Figure 2. In general, the "optimization" procedure must combine the experimental features of response surface methodology with the logical/numerical procedures of mathematical programming.

PROBLEM FORMULATIONS

We shall consider two basic approaches to formulating the problem of optimizing a multiple-variable, multiple-response simulation model. One approach is the familiar constrained optimization formulation in which one of the simulation responses, say η_1, is to be maximized or minimized, subject to maintaining the remaining m-1 responses within prescribed bounds. The second approach is the multiple-objective formulation, in which the m responses are either weighted to form a single objective or treated in a manner akin to goal programming. Each of these two formulations is described in the following sections.

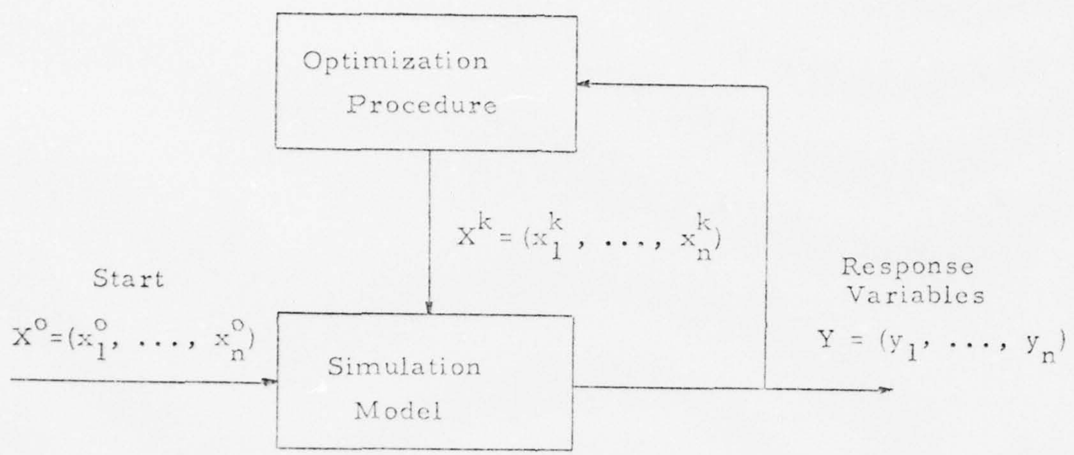


Figure 2. The interface between optimization and simulation

Constrained Optimization

Under the constrained optimization approach, the problem is stated as

$$\text{Max(or Min)} \quad \eta_1 = g_1(x_1,\ldots,x_n) \tag{10}$$

subject to the constraints

$$a_i \leq x_i \leq c_i, \quad i=1,\ldots,n \tag{11}$$

$$\eta_j = g_j(x_1,\ldots,x_n) \begin{Bmatrix} \geq \\ = \\ \leq \end{Bmatrix} d_j, \quad j=2,\ldots,m \tag{12}$$

The constraints expressed in (11) are bounds on the controllable input variables $x_1,\ldots,x_n$ and are typically known <u>a priori</u>. The bounds (11) generally form the known experimental region prior to conducting simulation trials. In contrast to that, the response functions $g_j(x_1,\ldots,x_n)$ in (12) are not usually known <u>a priori</u> and hence the responses η_j must be estimated experimentally via simulation. Thus, simulation trials performed at points satisfying (11) may yield responses violating (12).

To complicate matters even more, the random error ε_j can lead to erroneous decisions relative to feasibility with respect to the constraints in (12). The same is true relative to the objective function in (10). That is, one simulation trial can appear to represent an improvement over another when the true response at this particular set of values $x_1,\ldots,x_n$ does not. These difficulties can be countered through variance reduction techniques, as discussed by Fishman [12] and Kleijnen [19].

Multiple-Objective Optimization

One approach to a multiple-objective formulation is to assign weights $w_j, j=1,\ldots,m$ to the <u>m</u> responses and form a single objective function

$$\text{Max (or Min) } W = \sum_{j=1}^{m} w_j g_j(x_1,\ldots,x_n) \tag{13}$$

The bounds (11) still apply, so that the problem remains one of constrained optimization, but one in which the entire feasible region is known a priori. The weight $w_j, j=1,\ldots,m$ are typically assigned through the subjective judgment of the decision-maker in the system being simulated. These weights are usually normalized, so that

$$\sum_{j=1}^{m} w_j = 1 \tag{14}$$

One frequently encounters the situation in which certain of the responses η_j are to be maximized and other minimized. This case is handled by maximizing the negative of those functions which are to be minimized, so that the objective function in (13) is rearranged to the form

$$\text{Max } W = \sum_{j=1}^{s} w_j g_j(x_1,\ldots,x_n) - \sum_{j=s+1}^{m} w_j g_j(x_1,\ldots,x_n) \tag{15}$$

where s functions are maximized and m-s functions are minimized.

A second approach to the multiple-objective formulation is one which casts the objective function as a "utility function"; that is,

$$\text{Max } U[g_1(x_1,\ldots,x_n),\ldots,g_m(x_1\ldots,x_n)] \tag{16}$$

subject to the bounds in (11). The formulation in (15) is a special case of that in (16), in which $U[g_j(x_1,\ldots,x_n)]$ is a linear additive function. Montgomery and Bettencourt [21] discuss various formulations of the multiple objective optimization problem, as well as several approaches to its solution, and demonstrate its application to multiple-response simulation.

Another multiple-objective optimization formulation is that called

goal programming. This procedure is initiated by establishing a set of goals in terms of the m system responses. These goals are expressed as

$$G_j = g_j(x_1,\ldots,x_n),\ j=1,\ldots,m \tag{17}$$

Each goal must have an associated right-side value d_j; that is,

$$G_j = g_j(x_1,\ldots,x_n) \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} d_j,\ j=1,\ldots,m \tag{18}$$

With a slight modification, each goal can be expressed as an equality

$$G_j = g_j(x_1,\ldots,x_n) + n_j - p_j = d_j, j=1,\ldots,m \tag{19}$$

where n_j is a negative deviation from d_j, and p_j is a positive deviation. Either n_j or p_j must be zero in any given solution, and both could be zero. Next, each of the m goals G_j is assigned to a priority level P_k, $k=1,\ldots,\ell$, where P_1 represents the highest priority and P_ℓ the lowest. For any goal falling within a given priority level P_k, the decision-maker can weigh one goal relative to another. The final step in problem formulation is to combine these several levels of goals into an achievement function which has the form

$$A = \left\{P_1(\bar{n}_1,\bar{p}_1),\ P_2(\bar{n}_2,\bar{p}_2),\ldots,P_\ell(\bar{n}_\ell,\bar{p}_\ell)\right\} \tag{20}$$

This achievement function is simply an ordered ℓ-vector. Its structure is predicated on one of the following procedures for achieving the j-th goal:

(a) To equal or exceed d_j, minimize n_j

(b) To equal or be less than d_j, minimize p_j

(c) To equal d_j, minimize $(n_j + p_j)$

A solution $(x_1^*,\ldots,x_n^*)$ is considered optimal if the corresponding value A^* is the same as or preferred to any other value A. Therefore, the general

goal programming problem is to find $x_1,\ldots,x_n$ so as to minimize the ordered vector (20) such that the goals (19) are satisfied and

$$
\begin{aligned}
x_i &\geq 0, \quad i=1,\ldots,n \\
n_j &\geq 0, \quad j=1,\ldots,m \\
p_j &\geq 0, \quad j=1,\ldots,m
\end{aligned}
\tag{21}
$$

The functions $g_j(x_1,\ldots,x_n)$ in (19) are generally unknown, but are usually assumed to be nonlinear. Any technique proposed for solving this problem in the simulation domain must provide experimental estimates of these unknown functions, as well as a mathematical procedure for optimization. Moreover, the experimental observations are produced via simulation - each simulation trial at a point $x_1,\ldots,x_n$ produces $\underline{m}$ responses y_j=1,...,m. Biles [4] has described the application of nonlinear goal programming to the multiple-response simulation problem, based on techniques proposed by Ignizio [17].

OPTIMIZATION TECHNIQUES

Various procedures have been applied in combining optimization and simulation to seek the "optimum" solution to systems possessing a single response η. The multiple-response problem described here is complicated by the necessity to observe several responses at once, and to incorporate these values into the optimization technique. But many of the same techniques that have been applied successfully to the single-response problem can, with appropriate modifications, be extended to accommodate multiple responses. Moreover, these modified procedures are often applicable to more than one of the aforementioned formulations of the multiple-response problem.

The optimization procedures described below fall into three categories: (1) direct search techniques, (2) first-order response surface methods, and

(3) second-order response surface procedures. Although numerous techniques will be cited, only a few broadly stated procedures will be outlined here. It should be remembered that, although we may refer to "optimization" techniques, the classical notion of an "optimum" solution is inapplicable due to the presence of the sampling error ϵ_j associated with each response variable η_j. Rather we shall seek a solution which hopefully lies close to the true solution. In a more formal sense, we might state that we are to some degree confident, say 90%, that the true solution lies within some interval about our estimated solution.

Direct Search Methods

Direct search methods are those which, applied in a purely computational manner, do not require the use of derivatives. These methods progress through a sequence of points according to some algorithm. Typical of this class of optimization techniques are the pattern search algorithm by Hooke and Jeeves [14], sequential simplex search by Spendley, Hext and Himsworth [26], and the so-called "complex" search method by M. J. Box [8]. In general, these direct search procedures make rapid early progress toward an "optimum", but iterate laboriously as a solution is neared. This is particularly true in the presence of random error, as encountered in simulation.

Among the direct search techniques, Box's "complex" method [8] has been found to be the easily adapted to a multiple-response environment. It also performs better than any of the other direct search techniques in the face of random error and constraints. In fact, "complex" search is not at all complex, but derives its name from a contraction of the words "constrained simplex": it evolved from the sequential simplex method [26] and the necessity to deal with constraints. This report describes a modification of Box's method which makes it especially suitable for the multiple-response

simulation problem. The following procedure describes a generalized "complex" procedure as it might be applied to the multiple-response simulation problem:

1. Randomly generate a set of $N \geq n + 2$ search points $X^1,\ldots,X^N$ satisfying the known bounds (11).
2. Perform a simulation trial at each of these N search points and record the mN estimated responses y_j^ℓ, $j=1,\ldots,m$, $\ell-1,\ldots,N$.
3. Where a given point X^k is observed to violate one or more constraints, if such constraints apply with the particular problem formulation being employed, generate a replacement search point $X^{k'}$, perform a simulation trial at $X^{k'}$, and record the m estimated response at $X^{k'}$.
4. After N feasible search points have been established, evaluate the objective function for each of these N points. This "objective function" might be η_1 in (10), W in (15), U in (16), or A in (20). Among these N search points, find the worst point X^w; that is, the search point giving the least desirable value of the objective function. Define X^c as the centroid of the N-1 remaining points. Project from X^w through X^c to the image point $X^{w'}$. If the known bounds (11) are violated by this move, shorten the step to $X^{w'}$ until no violation occurs. Perform a simulation trial at $X^{w'}$.
5. Repeat steps 3 and 4 until a solution (X^*,Y^*) is obtained which represents the best solution that can be achieved within the available computer time. Figure 3 illustrates Box's complex search as applied to a constrained problem. Figure 4 gives a

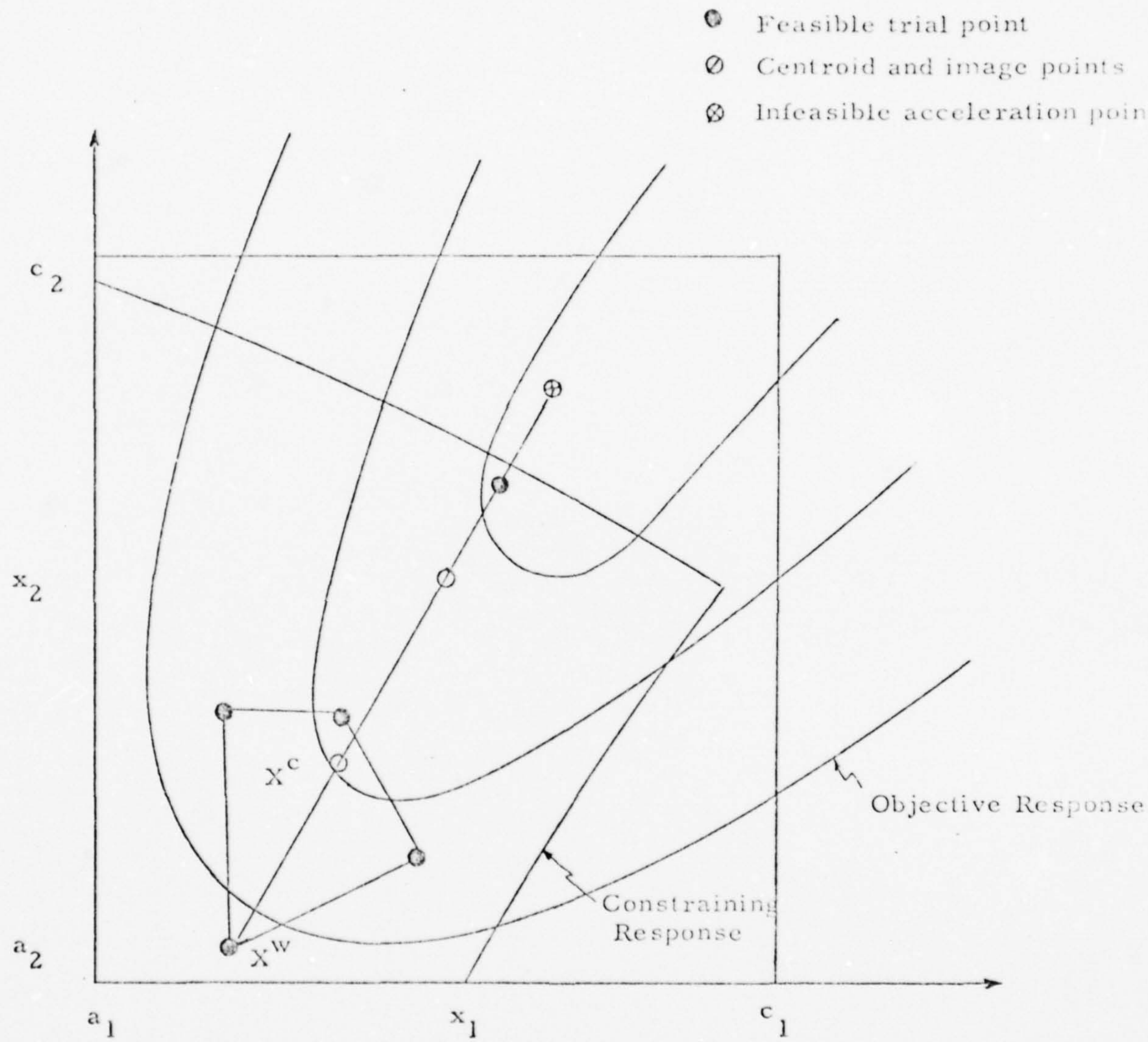


Figure 3. Complex Search

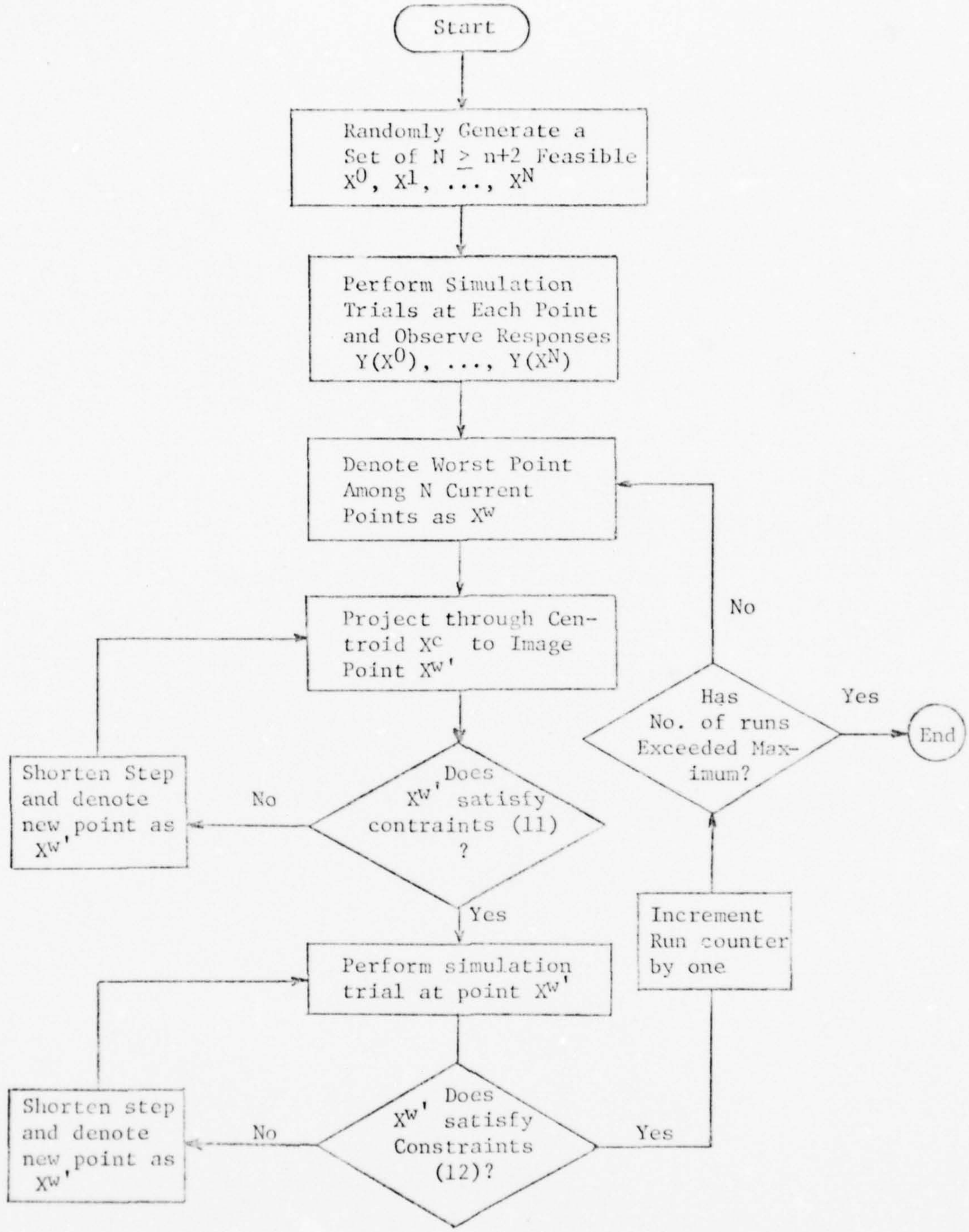


Figure 4. A multiple-response complex search procedure

a flow chart for a simulation/optimization method based on the procedure outlined above. Appendix A contains the program listing for a constrained optimization formulation of the Box complex search method.

A significant advantage of complex search is that, once an initial "complex" of N feasible simulation trials (and perhaps several infeasible trials) have been performed, trials are conducted one at a time thereafter. The search can be continued as long as improved solutions are obtained. If several successive simulations are performed at scattered points around the known experimental region without achieving an improved solution, however, the search can be terminated and the best available solution adopted.

First-Order Response Surface Methods

First-order response surface methods attempt to accomplish experimentally what the "method of steepest ascent" accomplishes computationally. From a current point X^k, a designed experiment is conducted (with a simulation trial at each design point) to estimate the gradient direction $\nabla g(X^k)$. Simulation trials are then conducted at points along this direction to a new point X^{k+1} which represents the best solution obtained along the direction $\nabla g(X^k)$. This process is an experimental approximation of

$$X^{k+1} = X^k + \lambda^k[\nabla g(X^k)] \tag{22}$$

The step length λ^k can be estimated by a line search or by a regression procedure as described by Biles [2,3].

The gradient direction $\nabla g(X^k)$ is estimated by placing an appropriate first-order experimental design, such as a 2^n factorial, 2^{n-p} fractional factorial, or n-dimensional simplex design (see Myers [22]) around the current point X^k. A simulation trial is performed at each point in the

selected experimental design. From these N observations the multiple linear regression model

$$\hat{y} = b_o + \sum_{i=1}^{n} b_i x_i \tag{23}$$

can be estimated (see Draper and Smith 10]). Since the gradient direction $\nabla g(X^k)$ is mathematically defined as the n-vector of first partial derivatives of $g(X)$ evaluated at X^k, it is clear that $\nabla g(X^k)$ is simply the n-vector of regression coefficients exclusive of the b_o term; that is,

$$\nabla g(X^k) = (b_1, \ldots, b_n)' \tag{24}$$

In the multiple-response simulation problem, a simulation trial is conducted at each design point in the selected first-order design and the $\underline{m}$ observations y_j^{ℓ}, $j=1,\ldots,m$ are recorded at each design point. Multiple linear regression is applied separately to each set of observations (assuming independence among the $\underline{m}$ responses), producing the $\underline{m}$ models

$$\hat{y}_j = b_{j,0} + \sum_{i=1}^{n} b_{j,i} x_i, \quad j=1,\ldots,m \tag{25}$$

and hence the $\underline{m}$ gradient vectors

$$\nabla g_j(X^k) = (b_{j,1}, \ldots, b_{j,n})', \quad j=1,\ldots,m \tag{26}$$

These estimates can then be employed in any one of several optimization schemes to produce an improved solution X^{k+1}. A generalized procedure for accomplishing this improved solution, and an estimated "optimum", will be described later. But first it is necessary to give attention to the experimental designs employed to estimate the gradient vectors $\nabla g_j(X^k)$, $j=1,\ldots,m$.

In selecting a first-order response surface design, it is usually desirable to minimize the variances of the regression coefficients $b_i, i=1,\ldots,n$. To accomplish this the first-order experimental design should be orthogonal.

That is, the placement of the N experimental points (in our case, simulation trials) is described by the N by n design matrix D, where

$$D = \begin{vmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ x_{1N} & x_{2N} & \cdots & x_{nN} \end{vmatrix} \tag{27}$$

Then an N by n+1 matrix X is constructed by placing a unit vector to the left of D. Thus,

$$X = \begin{vmatrix} 1 & x_{11} & x_{21} & \cdots & x_{n1} \\ 1 & x_{12} & x_{22} & \cdots & x_{n2} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ 1 & x_{1N} & x_{2N} & \cdots & x_{nM} \end{vmatrix} \tag{28}$$

It is usually convenient to code the design levels, so that the following conditions are achieved:

$$\left.\begin{aligned} \sum_{u=1}^{N} x_{iu}^2 &= N \\ \sum_{u=1}^{N} x_{iu} &= 0 \end{aligned}\right\} \quad i=1,\ldots,n \tag{29}$$

If the actual value of the u-th level of the i-th variable is z_{iu}, then the corresponding coded values is

$$x_{iu} = \frac{z_{iu} - z_{iu}}{S_i} \tag{30}$$

where

$$\bar{z}_i = \left(\sum_{u=1}^{N} z_{iu}\right)/N \tag{31}$$

and

$$S_i = \sum_{u=1}^{N} (z_{iu} - \bar{z}_i)^2/N \tag{32}$$

Then

$$X'X = \begin{vmatrix} N & 0 & 0 & \cdots & 0 \\ 0 & N & 0 & \cdots & 0 \\ \cdot & & & & \cdot \\ \cdot & & & & \cdot \\ 0 & 0 & 0 & \cdots & 0 \end{vmatrix} \tag{33}$$

Since the (n+1) - vector of regression coefficients $\bar{b}$ is estimated by the least squares relation

$$\bar{b} = (X'X)^{-1} X' \bar{y} \tag{34}$$

where $\bar{y}$ is the N-vector of response estimates obtained from N simulation trials, the variance of the regression coefficients b_i, $i=1,\ldots,n$ is given by

$$\mathrm{Var}(b_i) = \sigma^2/N, \qquad i=1,\ldots,n \tag{35}$$

where σ^2 is the variance of the error term ε. Since we are interested in m separate system response y_j, $j=1,\ldots,m$, equations (34) and (35) can be generalized to

$$\bar{b}_j = (X'X)^{-1} X' \bar{y}_j, \qquad j=1,\ldots,m \tag{36}$$

$$\mathrm{Var}(b_{ji}) = \sigma_j^2/N, \qquad \begin{array}{l} i=1,\ldots,n \\ j=1,\ldots,m \end{array} \tag{37}$$

For an orthogonal first-order design, the results in (33)-(37) hold, giving a so-called "minimum-variance" design. The 2^n factorial and 2^{n-p} fractional factorial designs are orthogonal and hence minimum variance. Orthogonal n-simplex designs can be easily constructed (see Myers [22]). Since n-simplex designs provide the minimum number of design points needed to estimate the multiple-linear regression models in (23) or (25), and are hence the most "economical" of the first-order response surface designs, they are especially attractive for the purpose proposed here. Brooks and Mickey [9] concluded that n-simplex designs offer the most efficient approach to estimating the gradient direction $\nabla g_j(X)$. Figure 5 illustrates 2^n factorial and n-simplex designs.

Biles [2,3] has described a first-order response surface procedure for approaching the constrained formulation of the multiple-response simulation problem. This procedure involves performing a first-order design around a current point X^k to estimate the gradient direction $\nabla g(X^k)$ according to relation (24). A line search is then performed along $\nabla g(X^k)$ to estimate an optimal step λ in (22). As long as the search remains interior to the region bounded by the constraints (11) or (12), the procedure is basically the same as that proposed by Box and Wilson [6]. If one or more constraints (11) or (12) are encountered, however, Biles [2,3] proposes that the gradient projection direction be followed (see Rosen [23]). The procedure for estimating the gradient projection direction is as follows.

Suppose that at an estimated boundary point X^k, $\underline{q}$ constraints are satisfied as equalities. These can be either the (11) or (12) constraints, or both. Let B_q be the n x q matrix of first partial derivatives of these active constraints. Thus, B_q consists of the $\underline{q}$ gradient vectors $\nabla g_j(X^k)$, j=1,...,q. That is,

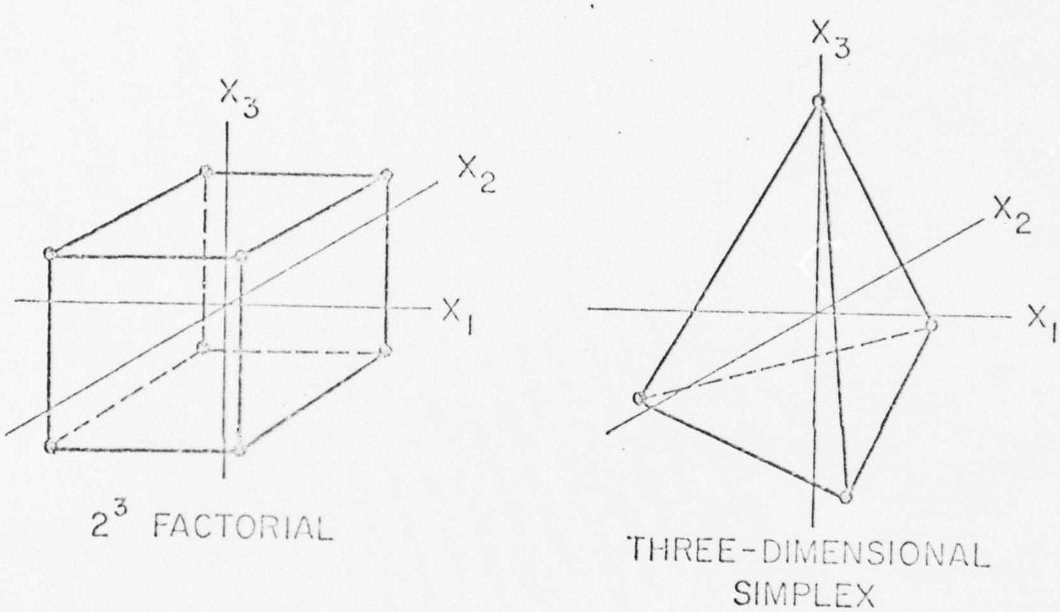


Figure 5. First-Order Response Surface Designs.

$$B_q = \begin{vmatrix} \partial g_1/\partial x_1 & \cdots & \partial g_q/\partial x_1 \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \partial q_1/\partial x_n & & \partial g_q/\partial x_n \end{vmatrix} \tag{38}$$

Since $g_j(X)$, $j=1,\ldots,q$ denotes the set of binding constraint functions (a constraint (11) or (12) is said to be "binding" if it is satisfied at the equality), for the moment let $f(X)$ represent the objective function. Then $\nabla f(X^k)$ and $\nabla g_i(X^k)$, $j=1,\ldots,q$ represent the gradient vectors of the objective and constraint functions, respectively, evaluated at the boundary point X^k.

Performing a first-order response surface experiment about the boundary point X^k yields estimates of the gradient vectors $\nabla f(X^k)$ and $\nabla g_i(X^k), j=1,\ldots,q$ in the form of the vectors of regression coefficients. (If a constraint of type (11) is included in the set of binding constraints, the gradient vector has the form $(0,0,\ldots,1,\ldots,0)'$, where all elements are zero except the i-th element which is one). Following the procedure outlined by Rosen [23], the gradient projection direction is given by

$$S^k = [\nabla f(X^k)] - B_q \, (B'_q B_q)^{-1} \, B'_q \, [\nabla f(X^k)] \tag{39}$$

A "golden section" line search is performed along direction S^k until either (a) a local "optimum" is found, or (b) other constraints are encountered. This new point is denoted X^{k+1}. This procedure is repeated until the gradient projection direction S^k is approximately zero. This point X^* is taken as a "constrained optimal" solution. Figure 5 illustrates the application of the gradient projection procedure to a constrained optimization problem.

Swain [27] has compared other first-order response surface techniques, including those of Klingman and Himmelblau [20] and Zoutendijk [28], to

Rosen's gradient projection method [23]. He found little difference among these techniques in terms of experimental requirements, and hence computer simulation time, but saw significant variability in computational requirements in order to use these algorithms. The Zoutendijk methods of feasible directions [28] require greater computational effort than the other procedures.

Biles [4] demonstrated both first-order and second-order approaches to a nonlinear goal programming formulation of the multiple-response simulation problem. These approaches are based on Ignizio's adaptation [17] of the method of Griffith and Steward [13] to goal programming and, like the constrained procedures mentioned previously, combine simulation, experimental design and mathematical programming.

The following generalized procedure is followed in employing a first-order response surface approach to the multiple-response simulation problem. The particular problem formulation and optimization procedure will govern the precise sequence of steps in implementing this procedure.

1. Identify the known experimental region $a_i \leq x_i \leq c_i$, $i=1,\ldots,n$. Selecting a starting point X^0 within this region. With X^0 as its center, array an orthogonal first-order response surface design within a selected design radius. Place $n_c = n/2 \geq 2$ points at the design center X^0 (coded as the $\bar{0}$ - vector).
2. Perform simulation trials at each of the N experimental design points and record the responses y_j^ℓ, $j=1,\ldots,m$; $\ell=1,\ldots,N$. Using multiple linear regression, fit linear models of the form (23) and (25).
3. Apply the appropriate mathematical programming technique to locate the next center point in the search.
4. Repeat steps 1-3 until an "optimum" solution is located. It may be appropriate to add design points to complete a second-order response surface design to test this optimum solution. The procedure for

accomplishing this is described in the next section.

Figure 6 illustrates a sequential first-order response surface method applied to a constrained system. Figure 7 gives a flow chart for a simulation/optimization method based on the procedure outlined above. Appendix B gives a program listing for this procedure.

Second-Order Response Surface Methods

A second-order response surface approach to the multiple-variable, multiple-response simulation problem consists of one or more repetitions of a two-stage procedure: (a) the execution of a computer simulation trial at each point in a second-order response surface experimental design covering the known region given by (11) and the use of multiple linear regression to fit second-order regression models to the resulting data; and (b) the application of a suitable mathematical programming procedure to obtain a solution to the problem formulated in (10)-(12), in (15) together with (11), in (16) together with (11), or in (19)-(21). In contrast to the first-order methods, in which the optimization procedure was part and parcel with the experimental procedure, these procedures are distinct and sequential in the proposed second-order approaches.

The first step in the second-order approach is to identify the range of each input variable. A safe strategy is to cover the entire known region $a_i \leq x_i \leq c_i$, $i=1,\ldots,n$ with the first (and possibly only) experimental design. If we let α_i denote the radius of the n-dimensional hypersphere within which the design points are contained, then

$$\alpha_i = (c_i - a_i)/2, \qquad i=1,\ldots,n \tag{40}$$

is effectively the maximum radius we could construct. It is convenient to adopt the coding convention expressed in (29) - (30), but choosing x_{iu} in such a way that α_i satisfies (40). Myers [22] describes this coding process.

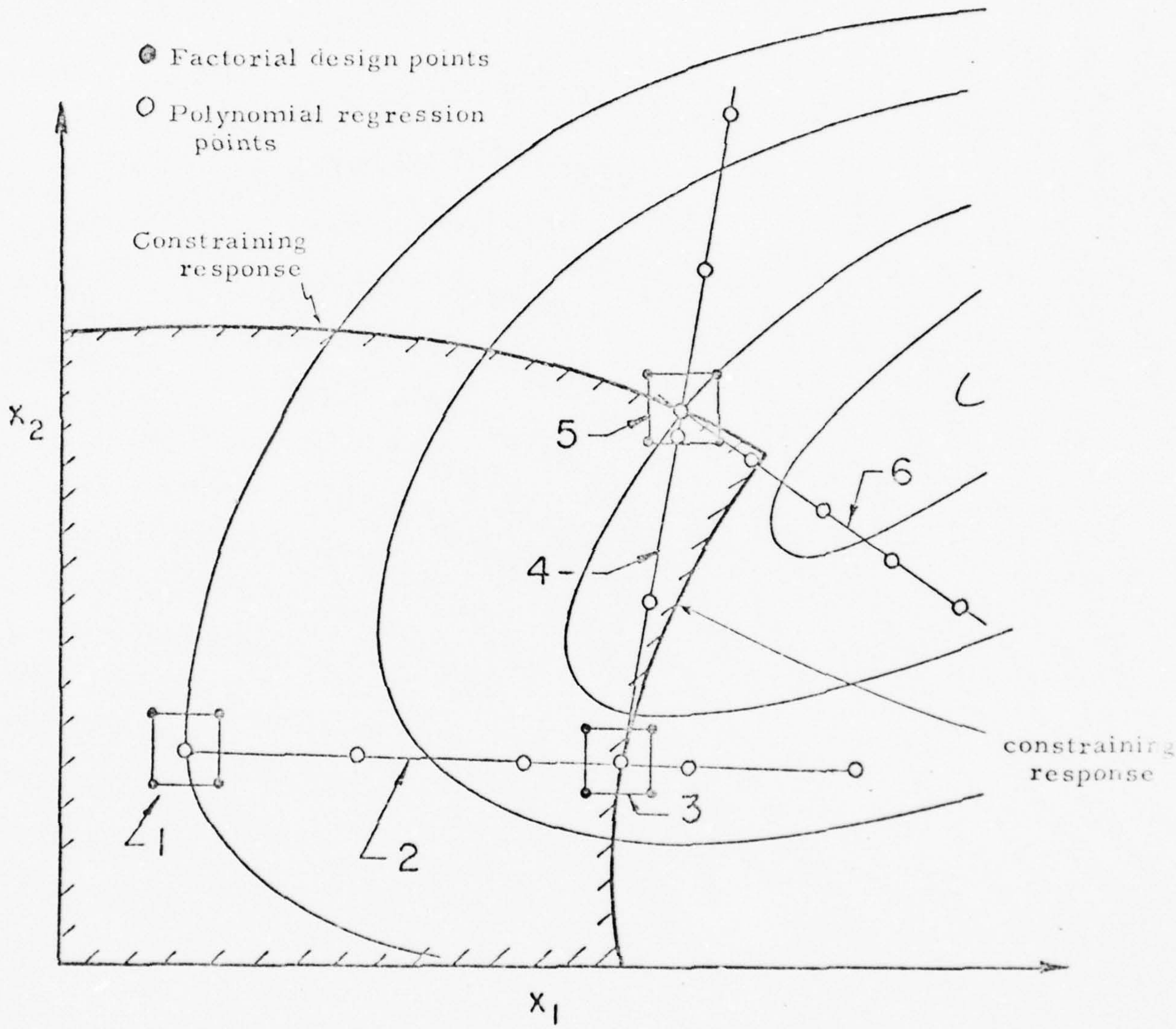


Figure 6. Gradient Projection Search.

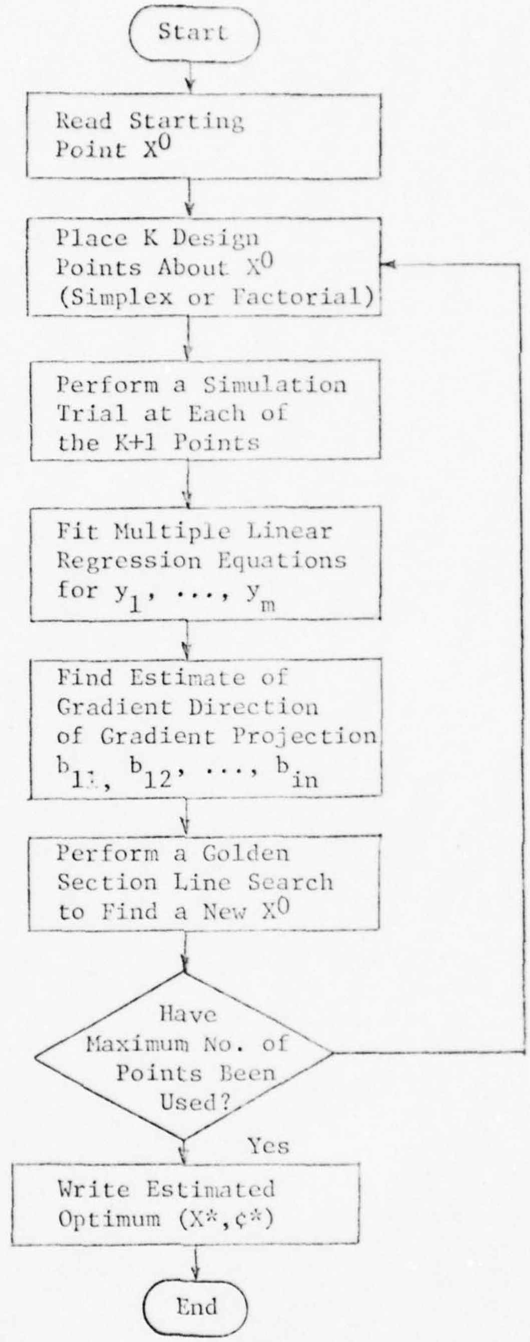


Figure 7. Flow Chart of Gradient Projection Search

The second-order fitted response surface has the form

$$\hat{y} = b_o + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} b_{ii} x_i^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} x_i x_j \tag{41}$$

where $\hat{y}$ is the estimate of the true response η at a given value $X = (x_1, \ldots, x_n)$ and the b_i and b_{ij} are regression coefficients in the fitted model. Since we must estimate $\underline{m}$ separate response relationships, equation (41) is modified to

$$\hat{y}_k = b_{k,o} + \sum_{i=1}^{n} b_{k,i} x_i + \sum_{i=1}^{n} b_{k,ii} x_i^2 + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} b_{k,ij}\, x_i x_j \tag{42}$$

$$k = 1, \ldots, m$$

Given the independence of the $\underline{m}$ responses, these m regression equations can be estimated independently from a set of $N \geq (n+1)\,(n+2)/2$ data points obtained by performing a simulation trial at each point in a second-order response surface design.

An experimental design employed for the purpose of estimating the regression coefficients in (42) must contain at least as many design points as there are coefficients b_i and b_{ij} in the fitted model, of which there are $(n+1)\,(n+2)/2$. Because of the non-linearity of (42), the experimental design must also have at least three levels of each controllable variable $x_i, i=1, \ldots, n$. It is also desirable to have a design which is rotatable; that is, the predicted response $\hat{y}$ at some point X is a function only of the distance from the design center to X and not a function of the direction.

The most widely used design for fitting a second-order model is the central composite design, shown in Figure 8 for n=2 and n=3. These designs consist of a 2^n factorial (or suitable fraction thereof), augmented by 2n axial points and k center points. A central composite design can be made

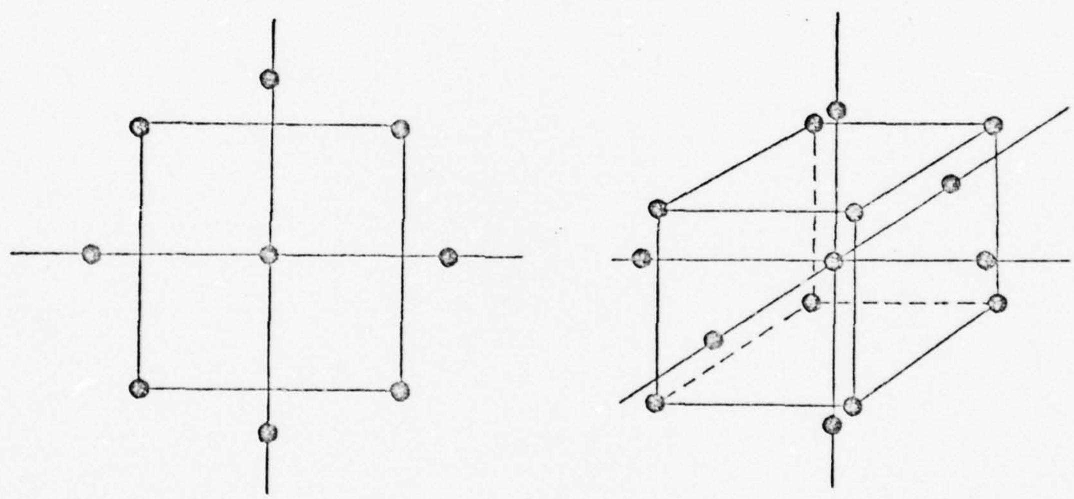


Figure 8. Central Composite Designs for Second-Order Response Surfaces.

rotatable by proper choice of α, the distance of the axial points from the design center. With the proper choice of the number of center points k, the central composite design can be made either orthogonal or uniform precision. Box and Hunter [7] give the characteristics of these designs for various sizes of $\underline{n}$.

Another important class of second-order response surface designs is the equiradial designs. That is, $N \geq (n+1)(n+2)/2$ design points are placed at points on an n-dimensional hypersphere. Figure 9 illustrates two equiradial designs for two variables. It is important to note that each of these designs is also equiangular, in that the N points are placed in such a way as to form N equal angles from the design center. Biles and Swain [5] have demonstrated the construction of such designs based on orthogonal n-simplex designs, thus achieving an efficient rotatable design.

Having estimated the $\underline{m}$ second-order regression equations (42) and formulated the appropriate optimization problem, it remains to apply mathematical programming to obtain a solution. For the constrained formulation, any of the following procedures could be employed: (a) Box's complex search [8]; (b) Rosen's gradient projection method [23]; or (c) one of Zoutendijk's methods of feasible directions [28]. For the weighted objective function formulation, these same three procedures are applicable. For the goal programming formulation, Ignizio's [17] procedure based on the method of Griffith and Stewart [13] is computationally efficient.

Figure 10 illustrates a complex search applied computationally to second-order response surfaces. Figure 11 gives a flow chart for a simulation/optimization method which employs the computational version of Box's complex search applied to second-order regression equations estimated from data obtained from simulation trials performed at each point in a second-order

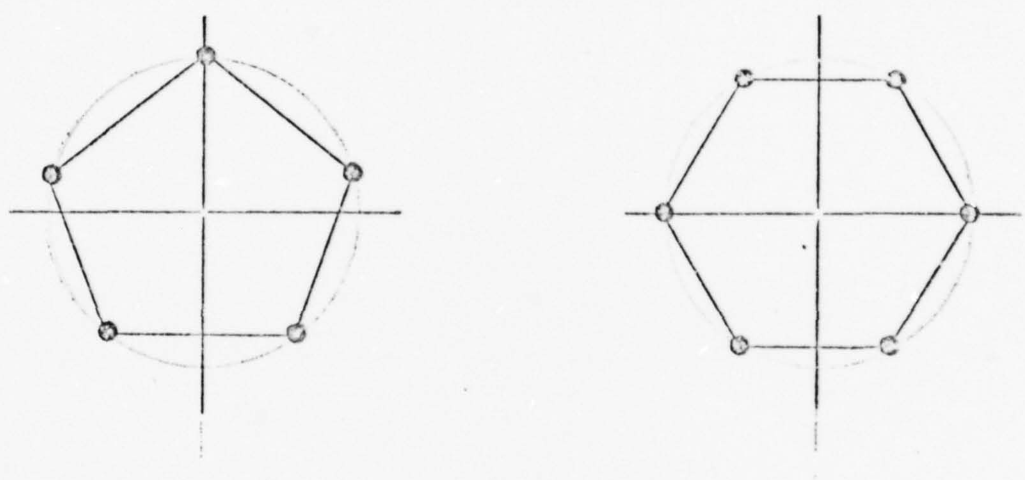


Figure 9. Equiradial Designs for Second-Order Response Surfaces.

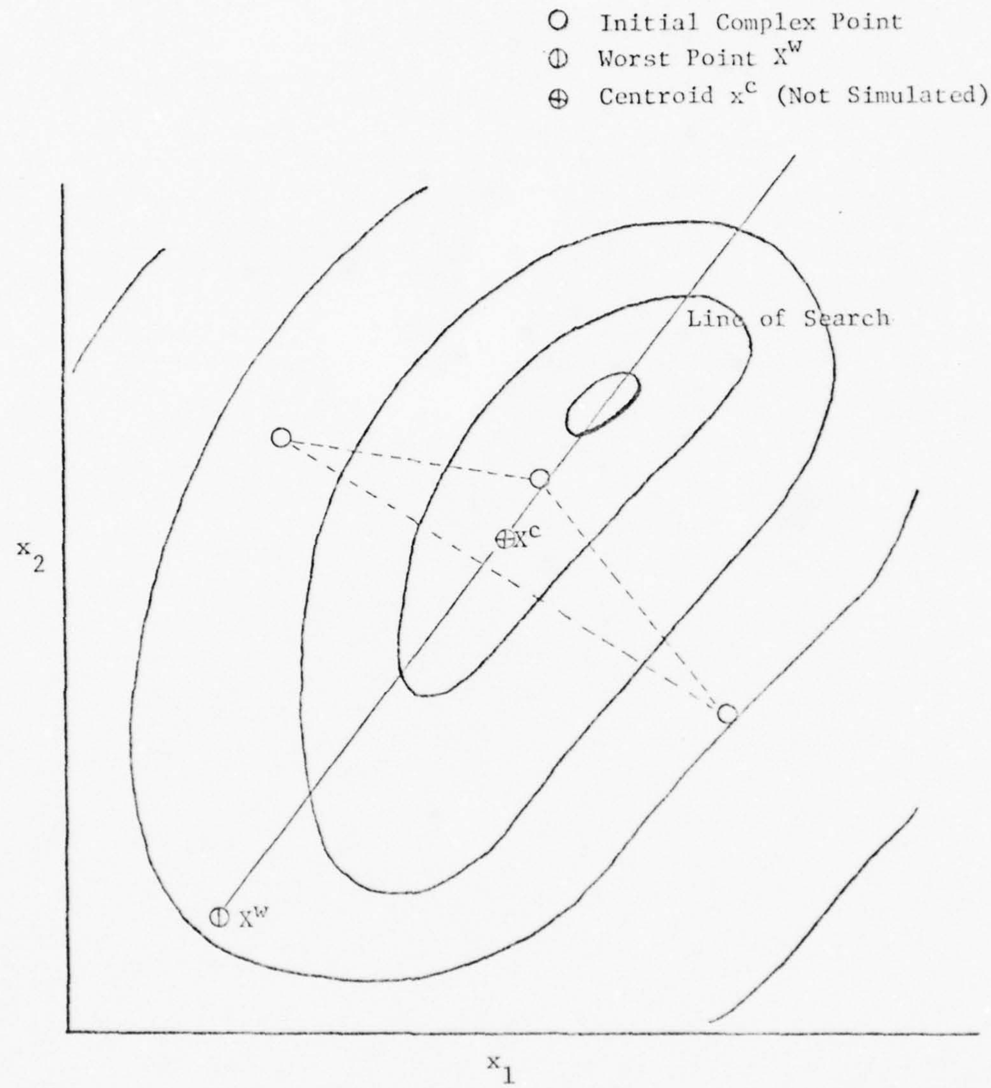


Figure 10. Complex Search with a Second-Order Surface.

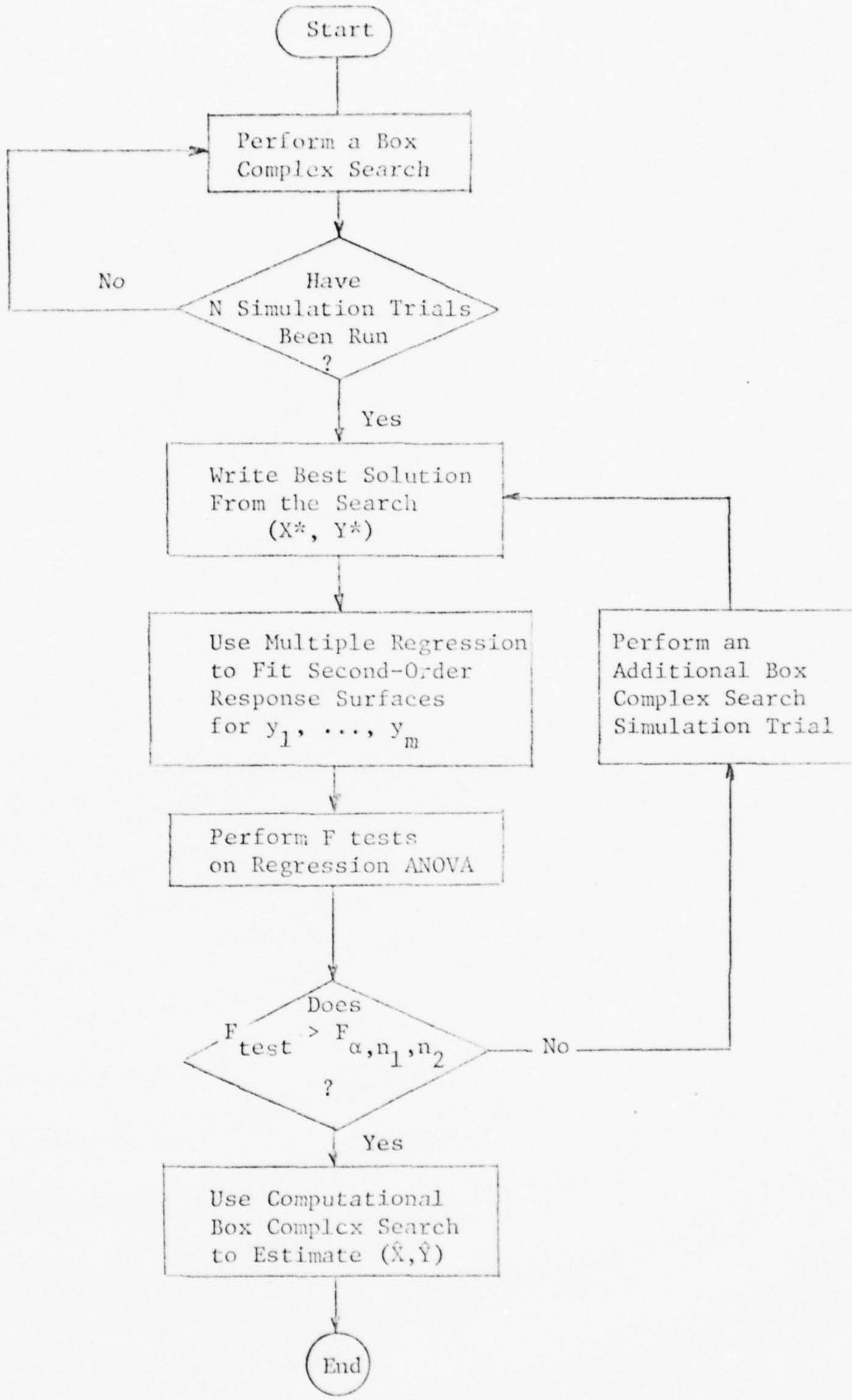


Figure 11. Flow-Chart of a Technique for Fitting Second-Order Response Surfaces to Complex Search Points

response surface design. Appendix C gives the program listing for this procedure.

Once an "optimal" solution has been obtained, it is necessary to compute a confidence interval, say at 90%, about the predicted "optimum." That is, we say that

$$P[y_{j\ell} \leq \eta_j^* \leq y_{ju}] = 0.9, \; j=1,\ldots,m \tag{43}$$

where $y_{j\ell}$ and y_{ju} are lower and upper bounds, respectively, on the 90% confidence interval for response η_j^* at X^*. Draper and Smith [10] give the procedure for computing this confidence interval. If the range $(y_{ju}-y_{j\ell})$ is found to be excessively large, it may be necessary to perform confirmation simulations in the vicinity of the predicted "optimum," or even an entire repetition of the second-order design in a reduced experimental region about the predicted "optimum."

Second-Order Response Surface Analysis of Box's Complex Search Experiments

Another approach to the simulation/optimization problem is to combine second-order response surface analysis with Box's complex search. Such a procedure would be applicable to any of the problem formulations described earlier. The general procedure for this method is as follows:

1. Perform a set of Box's complex search experiments as described in the first procedure.
2. Using the simulation responses at each search point, fit second-order regression equations.
3. Apply Box's complex search computationally to these equations.

The advantages of this two-phase procedure over either of its "component" techniques are that (1) it provides an objective stopping mechanism for the experimental Box complex search, and (2) it extracts latent information

about the unknown response functions $\eta_j(X)$, $j=1,\ldots,m$ that the search alone would not discover. The following is a description of this simulation/optimization procedure.

The Box's "complex" search phase of simulation experimentation proceeds as follows:

1. Randomly generate a set of $N \geq n + 2$ search points satisfying the known bounds $a_i \leq x_i \leq c_i$, $i=1,\ldots,n$.
2. Perform a simulation trial at each of these N search points X^k, $k=1,\ldots,N$ and record the mN simulated responses y_j^k, $j=1,\ldots,m$, $k=1,\ldots,N$.
3. Where a given search point X^k is found to violate a constraint of the form

$$y_j^k \cong \eta_j(X^k) \quad \begin{matrix} \geq \\ = \\ \leq \end{matrix} \quad d_j, \; j=1,\ldots,m \tag{6}$$

   generate a replacement point $X^{k'}$, perform a simulation trial at $X^{k'}$, and assess the feasibility of these new responses $y_j^{k'}$, $j=1,\ldots,m$. Repeat this step until exact N search points satisfying both the known bounds and any constrained responses (6) are obtained.
4. Identify as X^w that search point yielding the least desirable set of responses y_j^w, $j=1,\ldots,m$. Compute the centroid X^c of the N=1 remaining search points. Project X^w through the centroid X^c to a new point. If this point satisfies the known bounds, perform a simulation trial to obtain the response $y_j^{w'}$, $j=1,\ldots,m$ at $X^{w'}$. If any responses fail to satisfy the constrained responses (6), shorten the step from X^w through X^c, and repeat this step until a feasible new point $X^{w'}$ is obtained. This point replaces X^w in the "complex".

5. Repeat step 4 until either of the following conditions occurs:

   (a) an estimated solution is obtained at $(\hat{X},\hat{Y})$; or

   (b) a predetermined number of search points K (both those satisfying (6) and those not), is obtained, where $K \geq (n+1)(n+2)/2$.

When the complex search phase of simulation experimentation has been terminated, a best available, feasible solution $(\hat{X},\hat{Y})$ is retained for future reference. An examination is made of the arrangement of search points X^k, $k=1,\ldots,K$ in the known experimental region $a_i \leq x_i \leq c_i$, $i=1,\ldots,n$ to detect any regions that have a sparse density of search points. Any such region will have a search point placed at or near its center and a simulation trial performed there. This process guarantees coverage of the entire experimental region. It is also worthwhile to compute the centroid of the T total search points and to perform R replicates of a *simulation trial* (with different sets of initial random number seeds) at this centroid point. This total set of W=T+R simulation trials comprises a random experimental design in the factor space $a_i \leq x_i \leq c_i$, $i=1,\ldots,n$ with something approaching the minimum number of points for estimating second-order response surfaces. This minimum number of points is $(n+1)(n+2)/2$, corresponding to the number of regression coefficients in the model (42). Multiple linear regression is then employed to fit each of the m second-order surfaces to the W sets of responses y_j^k, $j=1,\ldots,m$, $k=1,\ldots,W$. A Box's complex search procedure is applied computationally to this system of m second-order response surfaces according to the particular problem formulation. The solution obtained by this procedure (X^*,Y^*) is compared to that obtained by complex search $(\hat{X},\hat{Y})$. If (X^*,Y^*) is preferred to $(\hat{X},\hat{Y})$ and X^* and $\hat{X}$ differ significantly, one or more confirmation simulation trials must be performed at (X^*,Y^*) and possibly

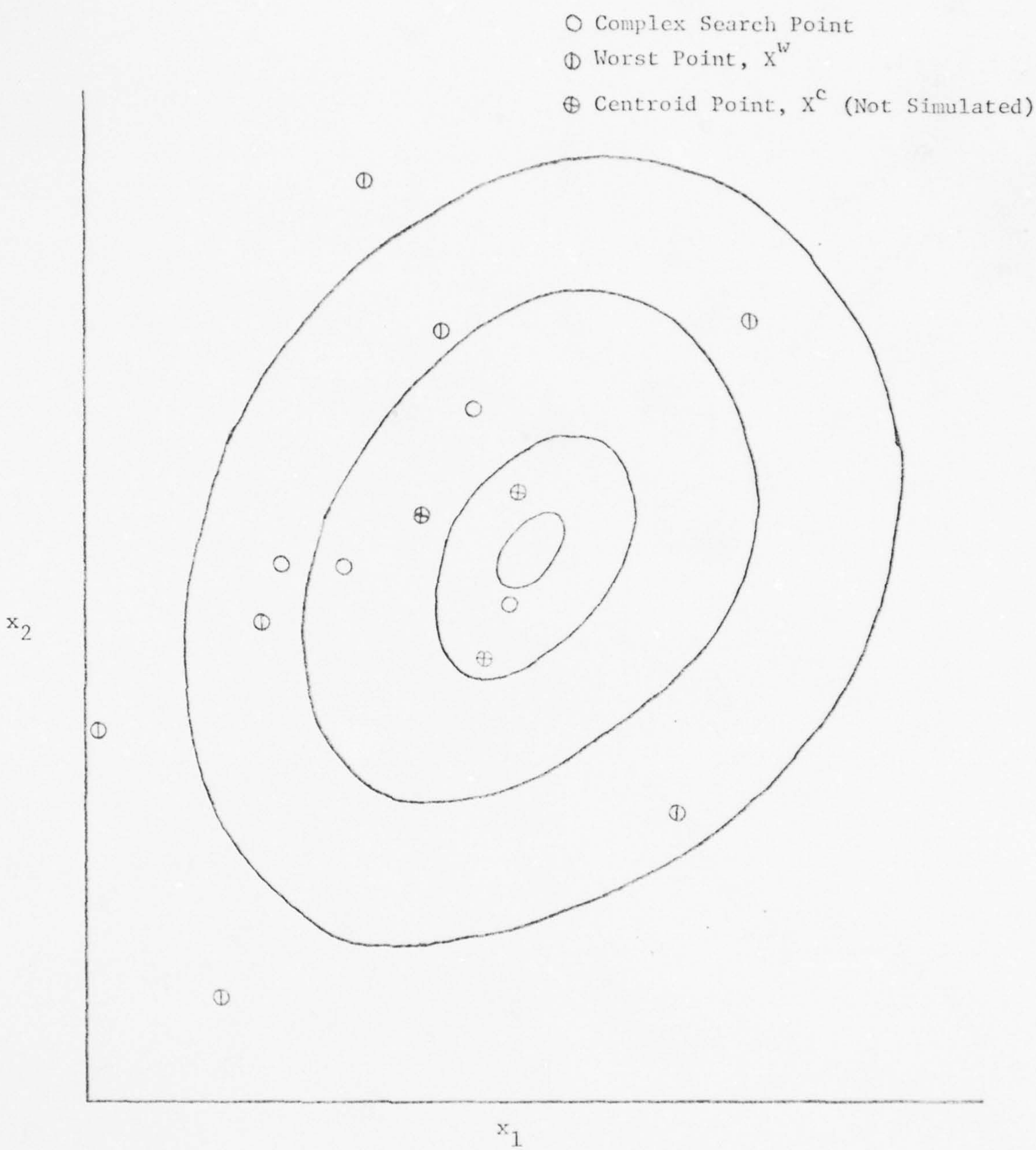


Figure 12. Second-Order Surface Fitted to a Sequence of Complex Search Points.

at $(\hat{X},\hat{Y})$ as well to enable a choice of the "optimum" solution.

Figure 12 illustrates a second-order response surface fitted to a set of points obtained via an experimental complex search, and then a computational complex search applied to this surface. Figure 11 gives a flow chart for this simulation/optimization procedure. Appendix D is a program listing of this procedure.

SIMULATION TEST MODELS

Three simulation test models were used to evaluate each of the four optimization methods. These simulation models are as follows:

1. A stochastic inventory system as described in Ignall [16] and Hunter and Naylor [15].
2. A tank duel model as described in Montgomery and Bettencourt [21].
3. A naval minefield evaluation model as described by Bailey and Weedon [1].

Exploratory work was performed but not completed on a fourth simulation model, the SPEARS anti-aircraft Naval defense model by Kaplan et al [18]. These models are discussed in detail in the following sections, but test results are discussed later in this report.

Stochastic Inventory System

Ignall [16] and Naylor and Hunter [15] describe the application of experimental design to the optimization of computer simulation responses. They employed as a simulation test model a discrete-event simulation of a stochastic inventory system in which mean daily demand and order lead time are random variables with known probability density (mass) functions. The lone simulation response was

y = mean daily cost, $

which was the sum of carrying, setup and shortage costs. The two controllable variables were

x_i = reorder point (ROP) ,

x_2 = economic order quantity (EOQ),

The optimization problem was

$$\text{minimize} \quad y = \eta(x_1, x_2)$$

subject to

$$-5 \leq x_1 \leq 90 \qquad\qquad 50 \leq x_2 \leq 250$$

Ignall [16] found as a solution $y = \$76$ at $x_1 = 45$ and $x_2 = 175$ units, for a given set of values for the several constants in the model.

The manner in which this simulation test model was utilized for the current research did not involve an actual simulation program. Rather, the 20 x 9 *response* table was used as the simulator. This response table is shown in Figure 13. For a given (x_1, x_2), an interpolation was performed in this response table to find y. This enabled a comparison of the four optimization procedures without introducing a confounding influence from the simulation model. It should also be noted that, since the "simulation" produced a single response y, this simulation test model did not fully test the optimization procedure but principally evaluated the interface between the simulator and optimization modules.

Tank Duel Model

In describing the application of multiple response surface methods in computer simulation, Montgomery and Bettencourt [21] employed a stochastic simulation model of a tank duel. The model simulates brief fire engagements between two armored vehicles. A stationary defending vehicle (Blue Tank)

y, AVERAGE DAILY COST, $

x_2, EOQ

x_1, ROP	50	75	100	125	150	175	200	225	250
-5	312	221	168	143	123	118	107	104	100
0	288	202	155	130	112	109	101	97	96
5	267	185	143	120	105	101	96	82	92
10	249	170	132	110	99	97	91	88	88
15	232	159	125	105	95	91	89	88	87
20	215	147	117	97	90	87	86	83	85
25	198	136	109	92	85	83	84	81	84
30	180	124	100	88	80	80	81	79	83
35	167	117	93	85	78	78	79	80	83
40	160	111	91	84	77	77	79	79	82
45	154	107	90	81	78	76	79	81	83
50	145	102	87	80	77	77	79	81	83
55	142	101	86	79	78	79	80	82	85
60	138	100	86	79	79	79	80	85	87
65	135	99	85	79	80	80	81	85	88
70	134	100	86	81	81	81	83	87	90
75	134	101	88	83	83	84	85	89	93
80	135	101	89	85	84	86	87	90	95
85	134	103	91	87	86	88	88	92	97
90	135	105	93	88	88	89	91	95	99

Figure 13. Response Table for Stochastic Inventory Simulation (Ignall [16]).

fires first at a fully-exposed attacking vehicle (Red Tank). The engagement ends when a kill occurs or a predetermined time limit of 120 seconds expires.

The input variables to the tank duel model are presented in detail in [21]. But for purposes of evaluating and comparing the four simulation/optimization procedures described in this report, the following independent variables x_i and response variables y_j were chosen:

x_1 = mean time to fire first round for the Blue crew (sec.),

x_2 = mean time between rounds (sec.),

y_1 = probability of Blue victory,

y_2 = expected number of rounds fired by the Blue tank.

The optimization problem was framed as one of constrained optimization, as follows:

$$\text{maximize} \quad y_1 = g_1(x_1, x_2)$$

$$\text{subject to} \qquad 5 \leq x_1 \leq 25 \qquad 5 \leq x_2 \leq 25 \qquad 0 \leq y_2 \leq 2$$

Mine Hunting [1]

This model, written in FORTRAN IV, is a Monte Carlo computer simulation which is used to evaluate mine hunting tactics. This program may be used either to evaluate minefields or, in conjunction with a sustained attrition minefield evaluation model, to compare the effectiveness of minesweeping to that of mine hunting. The basic operation is to move a mine hunting ship through a minefield. Each time the ship passes a mine or minelike object, its distance from that object is calculated, and a random number in the (0,1) range is drawn and compared with the probability of the ship's sonar detecting the object. If the object is detected, a similar procedure determines if the object is correctly classified and, in the case of mines, neutralized.

The ship's course is altered to avoid mines it has detected. Actuation and detonation of mines and damage of ships is simulated for both mine hunters and traffic ships.

Inputs include the number of mines, shape of minefields, density of bottom clutter, number of hunters, length of operation, effectiveness of sonar, condition of bottom, number of traffic ships, and method of hunting. Geographically, the model is limited to an area 20 miles square (a function of the number of mines) while the force limits are 100 hunters, 5 traffic ship types, 60 types of mines or minelike objects (e.g., bottom clutter), and 750 individual mines or minelike objects.

Outputs include mines detected, mines neutralized, ship damage, and threat of the minefield as a function of time.

Two separate problems were used to evaluate the four optimization procedures with the mine hunting model. In each of these problems, the optimization was sought from the standpoint of the force which deploys the minefield. That is, the objective was to maximize minefield effectiveness in terms of damage to ships which attempt to traverse the field.

Problem 1 used two mine types, arming delays and a total of 58 mines. Problem 2 used only one mine type, no arming delays and 61 mines. In Problem 1, all traffic ships (i.e., the target ships as opposed to the mine sweepers) attempted to transit the mine field during the period 119-120 hours, while in Problem 2 traffic was uniform over the entire 120 hour period.

Additional information about the problem that was common to both plans is as follows:

- The minefield was to cover an area one nautical mile wide and ten nautical miles long. (The input provided was for a 1800 ft. x .0 nm. field; this area required fewer mines, hence reduced running time. It was assumed that mine requirements were proportional to the area

of the minefield so that the plan for a 1nm. x 10 nm. area used (6076.1/1800) x the number of mines in the 1800' x 10 nm. area).

- Water depth was constant (hence for a given mine type/target type/target speed combination, the characteristics of a mine-target interaction were the same everywhere).
- The target ships travelled at three knots with a navigational error of 100 yards (i.e., the perpendicular distance from the target's actual location to the intended location was normally distributed with a mean of zero and a standard deviation of 100 yards).
- Two minesweepers were available and had a combined capability of making ten transits per day at a safe speed of five knots. They were assumed to have actuation characteristics identical to the target but were safe from damage.
- The mines were assumed to be uniformly randomly distributed throughout the minefield.

The objective of each plan was to provide a minefield with a 90 percent chance of damaging (mission about level) at least two of the ten target ships.

The simulation/optimization problem for each of these two minefield evaluation problems was formulated as follows:

x_1 = mean of the arming time distribution, where arming time was Poisson distributed; $6 \leq x_1 \leq 12$ hrs.

x_2 = mean of the mine count distribution, where mine count was Poisson distributed; $3 \leq x_3 \leq 5$ counts.

y_1 = average ship damage, ships (summed over all ship types)

y_j, $j = 2, \ldots, 6$ = average "threat" in period j-1, where period 1 is 0 - 24 hrs., period 2 is 24-48 hrs., etc.

A constrained optimization problem was

$\max y_1$

subject to $6 \leq x_1 \leq 12$; $3 \leq x_2 \leq 5$; $0 \leq y_j \leq 0.25$, $j = 2, \ldots, 6$.

The simulation/optimization interface for the minefield evaluation model involved the following:

- The main program was modified to become Subroutine SIMUL in the optimization program.
- In Subroutine SIMUL, each mine was assigned random arming time and a random ship count from Poisson distributions having parameters x_1 and x_2, respectively.
- In Subroutine AVGTHR, in which the damage to traffic ships is tallied and minefield "threat" is computed, special program segments were written to give values to the response variables y_j, $j = 1, \ldots, 6$.

Air Attack on a Surface Fleet [18]

This model, written in FORTRAN IV, evaluates antiair warfare tactics related to the following factors:

i. Force disposition - e.g., dispersed or integral formation concepts;

ii. Fire control doctrine - e.g., shoot, look, shoot;

iii. Command and control - e.g., coordinated or autonomous assignment doctrine;

iv. Cover and deception - e.g.,"look alike" ships and decoy forces;

v. Deployment of remote sensors - e.g., antielectronic warfare aircraft and radar picket ships;

The effects of enemy force factics evaluated include multidirectional, multialtitudinal attacks, use of electronic countermeasures and penetration aids, "roll back' attack tactics, and weapon loading and launch criteria.

The model may be described as a one-sided, event-store, Monte Carlo computer simulation. Enemy attack tactics are fully defined at the outset of each game situation by inputs to the raid generator, and the attack is conducted as defined regardless of the evolution of the battle. Thus, enemy attack tactics fall into the category of the "uncontrollable factors" z_k, $k = 1, \ldots, p$ in equation (1). The defense force, typically a carrier task force, includes a surveillance radar network (shipborne and airborne air search radars, both conventional periodic scan types, and multimode aray radars), a communications network for track passing and firepower coordination, and defensive weapon systems (primarily SAM batteries and their associated tracking radars and fire-control computers). Task force runs may be made to establish sensitivity to changes in ship configurations (alternate radars or weapon systems) or in ship dispositions and assignment doctrine. The model was designed to be, to the greatest extent practicable, machine and installation independent.

Input data to this model define the size, composition, disposition, and fire-control tactics for both the offense and the defense units. Enemy forces are most often composed of aircraft and submarines that launch antiship cruise missiles from stand-off ranges, although other forms of attack are possible (e.g., gravity bombs). The model has the capability of handling up to 31 defense units (ships and AEW aircraft) and up to 255 offense units (launch vehicles, stand-off jammers, cruise missiles, etc.). Numerous other limits define the composition of each major unit (e.g., radars pership, missile launchers per ship, magazine capacity per launcher, cruise missiles per launch platform, flight path legs per target). Certain input parameters select tactical options (e.g., single or dual missile salvos) and simulation algorithm options (e.g., simple or complex radar detection models).

A summary of the processed input values is printed at the outset of a game situation. This may optionally be followed by a detailed listing of

event occurences (of all types or of a selected subset and over the entire battle or during a specified time interval). A summary listing at the completion of play tabulates the performance and status of individual units at the end of the game; this listing includes factors such as average range of detection by surveillance radars, intercept ranges and kill probabilities achieved by defensive missile systems, number of enemy weapon penetrations (hits) suffered by each defense unit, and damage status of major ship components at the end of game play.

Although no simulation/optimization trials were conducted with this model, trial simulations were executed to characterize its performance. As with the minefield evaluation model, the "air attack on a surface fleet" model would require that the optimization interface be "custom-fitted" by developing suitable FORTRAN code.

DISCUSSION OF RESULTS

This section discusses the results obtained from simulation/optimization runs with the four optimization procedures in combination with each of the three simulation models. The discussion is presented in three segments, one for each simulation model.

Stochastic Inventory Model [15,16]

Table 1 presents the results of several optimization attempts with the stochastic inventory model. The complex search procedure and the second-order reponse surface approach of employing multiple regression to fit a quadratic model to a set of complex search points gave very similar results, yielding near-optimal solutions with about 7n simulation trials. The second-order response surface method employing a central composite design produced a predicted optimum at a point for which the actual solution was 4 percent removed from the true optimum, using only 9 simulation trials. The sequential first-order response surface approach, employing either a 2^2 factorial design or 3-vertex simplex design (each augmented by a centroid point), with a golden section search along the gradient direction, required about 20n simulation trials to yield near-optimal solutions. Among the four optimization methods evaluated with this model, only the first-order response surface method fared poorly.

It is notable that, when a second-order response surface is fitted to the 14 search points obtained from a Box complex search, the predicted optimum solution is actually slightly worse than the best solution obtained via the search itself.

Table 1. Summary of Simulation/Optimization Results for the Stochastic Inventory Model [15,16]

Optimization Method	Starting Seed	Estimated Solution x_1	x_2	y	Number of Trials
Complex Search	12471	50	177	$76.20	13
	21437	43	245	81.00	14
First-Order Factorial Design	17332	70	125	$79.00	40
First-Order Simplex Design	17332	42	159	$76.78	41
Second-Order Central Composite Design	35188	53	183	$70.32 (Predicted) $79.00 (Actual)	9
Second-Order Complex Search	14271	41	163	$76.78 (Search) 77.52 (Predicted)	14

Notes: (1) Known Solution $x_1 = 45$ $x_2 = 175$ $y = \$76.00$

(2) CPU times less than 5 sec. per optimization run on an IBM 370/168 computer.

(3) Complex search terminated because $2n = 4$ search points were conducted without an improved solution.

Tank Duel Model [21]

Table 2 presents the results of four simulation/optimization runs with the tank duel model. An optimum solution lies near (x_1 = 8.2, x_2 = 12.5), Producing a probability of Blue Victory of 0.61 with an expected firing of 2 Blue rounds. Thus, the "expected rounds fired" response appears to bound the solution. As with the stochastic inventory model, only the sequential first-order response surface approach produces unacceptably costly experimentation.

In contrast to the result with the stochastic inventory model, the predicted solution obtained after fitting second-order response surfaces to the two sets of responses obtained in the Box complex search represents a significant improvement over the best point observed in the search.

Mine Hunting Model [1]

As seen in Table 3, only three optimization methods were evaluated with the mine hunting simulation model. The complex search method, as with the previous models, was terminated after 2n simulation trials had been performed without obtaining an improved solution. For example, with problem 1, the estimated optimum solution was observed with search point 12, and since four more search points failed to improve on that result the search was terminated after point 16.

Focusing on the results of the second-order response surface method, employing a central composite design with 16 simulation trials, a solution x_1 = 10.6 hours mean arming delay per mine and x_2 = 3.1 ships mean ship count per mine results in an expected kill of 0.51 ships. The threat profile is essentially zero during the first 72 hours of minefield operation, and increases to levels between 0.15 and 0.2 during the last 48 hours of operation when enemy ship traffic actually attempts to traverse the field.

Table 2. Summary of Simulation/Optimization Results for the Tank Duel Model [21]

Optimization Method	Estimated Solution x_1	x_2	y_1	y_2		Number of Trials
Complex Search	12.9	10.1	0.59	1.97		20
First-Order Simplex Design	8.2	12.4	0.60	1.97		55
Second-Order Central Composite	8.0	12.5	0.61	2.00		9
Second-Order Complex Search	13.5 8.2	11.3 12.2	0.57 0.61	1.86 1.99	(Search) (Predicted)	15

Notes:

(1) Known optimum at $x_1 = 8.2$ sec., $x_2 = 12.5$ sec., $y_1 = 0.61$, $y_2 = 2$ rds.

(2) CPU times less than 5 sec. per optimization run on an IBM 370/168 computer.

(3) Complex search was terminated because $2n = 4$ search points were evaluated without obtaining an improved solution.

Table 3. Summary of Simulation/Optimization Results for the Minefield Evaluation Model [1]

Optimization Method	Problem Number	Estimated Solution								Number of Trials
		x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	
Complex Search	1	11.4	3.9	0.43	0.000	0.000	0.012	0.014	0.030	16
	2	6.2	3.5	0.32	0.000	0.027	0.027	0.044	0.058	9
Second-Order Central Composite	1	10.6	3.1	0.51	0.000	0.000	0.000	0.183	0.163	16
	2	6.0	3.0	0.37	0.000	0.005	0.044	0.091	0.128	16
Second-Order Complex Search	1 (Search)	8.6	3.6	0.55	0.000	0.000	0.000	0.183	0.140	25
	1 (Predicted)	8.8	4.0	0.26	0.000	0.000	0.000	0.153	0.129	

Notes: (1) True optimum solution is unknown.

(2) In each simulation trial, 5 replicates of the simulation were averaged.

(3) Optimization runs required about 10 minutes of CPU time each on an IBM 370/168 computer.

(4) The sequential first-order response surface method was not evaluated due to excessive computer time.

(5) The second-order complex search procedure did not coverge in a 10-minute run.

This result points out a difficulty in interfacing an optimization procedure with a simulation model when a gaming strategy is involved. In a sense, the optimization procedure "learns" the enemy tactic as the series of simulation runs progresses. In this instance, the optimization procedure "sees" that traffic ships traverse the field late in the game. To counter this "learning" behavior, it is necessary to level the threat over the entire period of minefield operation. Since such a narrow feasible region would cause complex search to yield a high proportion of infeasible responses, the preferred optimization method would be a second-order response surface approach, either by a designed experiment or by fitting a quadratic surface to Box's complex search results.

SUMMARY AND CONCLUSIONS

This research has investigated four alternative approaches to optimization of simulated systems having multiple independent variables x_i, i=1, ..., n and multiple simulation responses y_j, j=1, ..., m. Of these four methods, Box's complex search [8], a second-order response surface approach employing a central composite experimental design [6,7] followed by a computational complex search, and fitting a second-order response surface to a succession of complex search points, all yield "good" solutions in an "economical" number of computer simulation trials. The sequential first-order response surface method converged to near-optional solutions, but required an excessive number of simulation trials. This excessive number of trials derived from the golden section line search along the estimated gradient direction. Had a polynomial regression line search employing, say, five trials been conducted as in Biles [2,3], two complete cycles of gradient-determining blocks and line searches could have been performed in about 8n to 10n trials.

The most crucial aspects of the experimental Box complex search approach to simulation/optimization are twofold:

1. A termination criterion based on either (a) a maximum number of simulation trials or (b) a certain number of simulation trials without obtaining an improved solution. The typical criterion used to terminate a computational Box complex search is convergence; such as $y_{k+1} - y_k \leq \varepsilon$. Such a criterion can lead to excessive simulation trials in the experimental mode, however, unless ε is coarse.

2. In an experimental region in which the implicit constraints on simulation responses $y_2, \ldots, y_m$ describe a small feasible region, a number of infeasible simulation trials will be conducted. In fact, the probability that an early simulation trial is feasible with respect to implicit constraints is $P_f = R(Y_f)/R(E)$, where

   P_f = probability of a feasible trial

   $R(Y_f)$ = region subtended by the implicit constraints

   $R(E)$ = region subtended by the explicit constraints

   Once n+2 feasible simulation trials have been found, and complex search begins to seek improved solutions, there is a higher probability of obtaining a feasible simulation trial. As the "complex" becomes tighter, this probability approaches one. But the possibility of a large number of infeasible trials early in the search is perplexing.

But the capability of fitting quadratic response surfaces to a set of at least (n+1)(n+2)/2 complex search points (this being the number of regression coefficients in the quadratic model) in an "incomplete" complex search overcomes this difficulty.

The second-order response surface approach has the advantage of requiring relatively few simulation trials, but the accuracy of the predicted optimum solution $(\hat{X},\hat{Y})$ depends on how well the true surfaces are represented with quadratic estimators. In those cases where there is some knowledge of the nature of the true surfaces, the modeler is better able to hypothesize the fitted model (which may not be quadratic). Although the central composite design has superior statistical properties to the purely random design, such as obtained from a set of complex search points, the complex search approach will usually place more than one simulation trial in the neighborhood of the predicted optimum. This affords a smaller confidence region about the predicted optimum.

The computer programs listed in Appendices A through D offer a means of interfacing these optimization procedures with simulation models of interest. Minor alterations in the optimization program would be necessary for a particular simulation model, but major modifications in the simulation model itself could be entailed. In general, one could expect to have to "custom fit" the optimization program to a given simulation model. this is especially true of the typical naval simulation model, because modelers tend to "start from scratch" in developing models for their own needs.

This research has provoked two recommendations relative to naval simulation modeling activities in the several naval laboratories:

1. There is a definite need to have naval simulationists apply up-to-date statistical methodology, particularly variance reduction, in their modeling efforts.
2. There is a need to have naval simulationists become familiar with general simulation languages such as GASP-IV and SIMSCRIPT 2.5. It is also apparent that certain specialized modeling techniques, such as Q-GERT, is vastly under utilized in naval operations modeling.

Developing FORTRAN simulation models from "scratch" is sometimes necessary, but very often considerable time, effort and expense is spent in duplicating the kinds of functional capability contained in available and tested simulation soltware.

The most pressing need in terms of continuing research in the area of optimization of naval simulation models is to incorporate the techniques of game theory into the optimization approach. So many naval models embody a "friendly force/enemy force" focus, with decision-making capability on both sides, that optimization must entail game theory considerations.

REFERENCES

1. Bailey, J. L., and M. A. Weedon, "The Minefield Analysis with Hunting Evaluation Model (MAYHEM)," NWL Technical Report TR-2742, U. S. Naval Weapons Laboratory, Dahlgren, Virginia, May 1972.

2. Biles, W. E., "A Gradient-Regression Search Procedure for Simulation Experimentation," Proceedings of the 1974 Winter Simulation Conference, Washington, D.C., 1974, p. 491-497.

3. Biles, W. E., "Response Surface Methods for Optimization of Constrained Multivariable Processes," Industrial and Engineering Chemistry: Process Design and Development, Vol. 14, No. 2, 1975, p. 152-158.

4. Biles, W. E., "Optimization of Multiple-Objective Computer Simulations: A Nonlinear Goal Programming Approach," Proceedings of the 1977 AIDS Conference, Chicago, 1977.

5. Biles, W. E., and J. J. Swain, "Strategies for Optimization of Multiple Response Simulation Models," Proceedings of the 1977 Winter Simulation Conference, Gaithersburg, Maryland, December 1977.

6. Box, G. E. P., and K. B. Wilson, "On the Experimental Attainment of Optimum Conditions," Journal of the Royal Statistical Society, B, Vol. 13, 1951, p. 1-45.

7. Box, G. E. P., and J. S. Hunter, "Multifactor Experimental Designs for Exploring Response Surfaces," Annals of Mathematical Statistics, Vol. 28, 1957, p. 192-242.

8. Box, M. J., "A New Method of Constrained Optimization and a Comparison with Other Methods," Computer Journal, Vol. 8, 1965.

9. Brooks, S. H., and M. R. Mickey, "Optimization Estimation of Gradient Direction in Steepest Ascent Experiments," Biometrics, Vol. 17, No. 1, 1961.

10. Draper, N. R., and H. Smith, Applied Regression Analysis, John Wiley and Sons, New York, 1966.

11. Dyer, J. S., "Interactive Goal Programming," Management Science, Vol. 19, No. 1, 1973.

12. Fishman, G. S., Concepts and Methods in Discrete Event Digital Simulation, John Wiley and Sons, New York, 1973.

13. Griffith, R. E. and R. A. Stewart, "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems, Management Science, Vol. 7, 1961.

14. Hooke, R. and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," Journal of the Association of Computing Machinery, Vol. 8, 1961.

15. Hunter, J. S., and T. H. Naylor, "Experimental Designs for Computer Simulation Experiments," Management Science, Vol. 16, No. , 1970, pp. 422-434.

16. Ignall, E. J., "On Experimental Designs for Computer Simulation Experiments," Management Science, Vol. 18, No. 7, March 1972, pp. 384-388.

17. Ignizio, J. P., Goal Programming and Extensions, Lexington Books, D.C. Heath and Company, Lexington, Mass, 1976.

18. Kaplan, D. J., L. C. Davis, M. E. B. Owens II, O. F. Forsyth and J.J. Penick, "SPEARS, an AAW Performance Simulation," NRL Report 7958, Naval Research Laboratory, Washington, D. C., June 1976.

19. Kleijnen, J. P. C., Statistical Techniques in Simulation, Parts I and II, Marcel Dekker, New York, 1975.

20. Klingman, W. R., and D. M. Himmelblau, "Nonlinear Programming with the Aid of the Multiple Gradient Summation Technique," Journal of the Association of Computing Machinery, Vol. 11, 1964.

21. Montgomery, D. C., and V. M. Bettencourt, "Multiple Response Surface Methods in Computer Simulation," Simulation, October, 1977, pp. 113-121.

22. Myers, R. H., Response Surface Methodology, Allyn and Bacon, Boston, Mass., 1971.

23. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming: Part II - Nonlinear Constraints," Journal of the Society of Nonlinear and Applied Mathematics, Vol. 8, 1961.

24. Smith, D. E., "Requirements of an Optimizer for Computer Simulations," Naval Research Logistics Quarterly, Vol. 20, No. 1, 1973.

25. Smith, D. E., "An Empirical Investigation of Optimum-Seeking in the Computer Simulation Situation," Operations Research, Vol. 21, 1973, pp. 475-497.

26. Spendley, W., and G. R. Hext, and R. F., Himsworth, "Sequential Application of Simplex Designs in Optimization and Evolutionary Operations," Technometrics, Vol. 4, 1962.

27. Swain, J. J., "Efficient Strategies for Constrained Experimental Optimization," unpublished Master's thesis, University of Notre Dame, 1977.

28. Zoutendijk, G., Methods of Feasible Directions, Elsevier, New York, 1960.

APPENDIX A

PROGRAM LISTING

FOR AN

EXPERIMENTAL BOX COMPLEX SEARCH

```
C
C [illegible]
C
      DIMENSION X(30,40),XP(30),C(30),WP(30),Z(20,30),Y(20),Z1(20)
      COMMON N,M,NR
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE2/ISEED1,ISEED2
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BLEE4/ITER
C
      READ (9,*) N,NP
      READ (9,*) (XL(I), I=1,N), (XU(I), I=1,N)
      READ (9,*) (XLP(I),I=1,N), (XUP(I), I=1,N)
      READ (9,*) YL,YU
      READ (9,*) SMALL
      READ (9,*) ICHECK
      READ (9,*) IX
      READ (9,*) ISEED1,ISEED2
C
      WRITE (12,1000)
      WRITE (12,*) N,NP
      WRITE (12,*) (XL(I), I=1,N), (XU(I), I=1,N)
      WRITE (12,*) (XLP(I), I=1,N), (XUP(I),I=1,N)
      WRITE (12,*) YL,YU
      WRITE (12,*) SMALL
      WRITE (12,*) ICHECK
      WRITE (12,*) IX
      WRITE (12,*) ISEED1,ISEED2
C
      ITER=0
      M=1+NP
      AM=M
      AMM1=M-1
C
      WRITE (12,160)
      DO 2 J=1,M
   10 DO 1 I=1,N
      CALL RANDU(IX,IY,RNUM)
      IX=IY
      DA=XUP(I)-XLP(I)
      X(I,J)=XLP(I)+RNUM*DA
    1 XP(I)=X(I,J)
      CALL FSUBPT(XP,IOUT,Y)
      ITER=ITER+1
      IF (IOUT.EQ.1) GO TO 105
      DO 101 I=1,N
  101 WRITE (12,170) I,XP(I)
      DO 102 I=1,NR
  102 WRITE (12,180) I,Y(I)
      GO TO 115
  105 DO 111 I=1,N
  111 WRITE (12,175) I,XP(I)
      DO 112 I=1,NR
  112 WRITE (12,185) I,Y(I)
      WRITE (12,130) ITER
      GO TO 10
  115 DO 15 L=1,NR
      Z(L,J)=Y(L)
   15 Z1(J)=Y(1)
```

```
   12 WRITE (12,120) ITER
      CONTINUE
C
      CALL ZOPT(KP,ZMAX,Z1)
   20 CALL WORST(K,ZMIN,Z1)
      DO 33 I=1,N
      SUM=-X(I,K)
      DO 30 L=1,M
   30 SUM=SUM+X(I,L)
   33 C(I)=SUM/AM1
C
      R=1.
   35 S=1.+R
      DO 37 II=1,N
   37 XP(II)=X(II,K)+S*(C(II)-X(II,K))
      CALL FSOPT(XP,IOUT,Y)
      R=R*0.5
      IF (IOUT.EQ.2) GO TO 35
      ITER=ITER+1
      IF (IOUT.EQ.1) GO TO 385
      DO 371 I=1,N
  371 WRITE (12,170) I,XP(I)
      DO 372 I=1,NR
  372 WRITE (12,180) I,Y(I)
      GO TO 400
  385 DO 391 I=1,N
  391 WRITE (12,175) I,XP(I)
      DO 392 I=1,NR
  392 WRITE (12,185) I,Y(I)
      WRITE (12,135) ITER
      GO TO 35
C
  400 IF(Y(1).LE.ZMIN) WRITE (12,145)
      IF (Y(1).LE.ZMIN) GO TO 35
      WRITE (12,140) ITER
      DO 50 I=1,N
   50 X(I,K)=XP(I)
      DO 55 L=1,NR
   55 Z(L,K)=Y(L)
      Z1(K)=Y(1)
      IF (ZMAX.GE.Z1(K)) WRITE (12,155)
      IF (ZMAX.GE.Z1(K)) GO TO 20
      WRITE (12,150)
      KP=K
      ZMAX=Z(1,KP)
      DO 100 I=1,M
      A=ABS(ZMAX-Z1(I))
      IF (A.GT.SMALL) GO TO 20
  100 CONTINUE
CC
      IF (ICHECK.EQ.0) GO TO 60
      DO 550 I=1,N
      SUM=X(I,1)
      DO 530 L=2,M
  530 SUM=SUM+X(I,L)
  550 XP(I)=SUM/AM
C
      CALL SIMUL(XP,Y)
      REWIND 10
```

```
      ITER=ITER+1
      DO 90 I=1,N
   90 WRITE (12,170) I,XP(I)
      DO 95 I=1,NR
   95 WRITE (12,180) I,Y(I)
      WRITE (12,200) ITER
C
      DO 590 I=1,M
      A=Y(I)-Z1(I)
      IF (A.GT.SMALL) GO TO 600
  590 CONTINUE
      GO TO 60
C
  600 CALL WORST(K,ZMIN,Z1)
      IF (Y(1).LE.Z1(K)) GO TO 60
      DO 610 I=1,N
  610 X(I,K)=XP(I)
      DO 630 I=1,NR
  630 Z(I,K)=Y(I)
      Z1(K)=Y(1)
      IF (ZMAX.GE.Z1(K)) WRITE (12,155)
      IF (ZMAX.GE.Z1(K)) GO TO 20
      WRITE (12,150)
      KP=K
      ZMAX=Z1(K)
      GO TO 20
CC
   60 WRITE (12,160)
      WRITE (12,165)
      DO 70 I=1,N
   70 WRITE(6,170) I,X(I,KP)
      DO 80 I=1,NR
   80 WRITE(6,180) I,Z(I,KP)
C
  120 FORMAT (/1X,I5,'  TH  ITERATION'/5X,'*****  THE ABOVE POINT IS AN
     &INITIAL POINT  *****'/)
  130 FORMAT(/66X,I5,'  TH  ITERATION'/65X,'*****  THE ABOVE POINT IS NO
     &T AN INITIAL POINT  *****'/)
  135 FORMAT(/66X,I5,'  TH  ITERATION'/65X,'*****  THE ABOVE POINT IS NO
     &T A FEASIBLE POINT  *****'/)
  140 FORMAT (/1X,I5,'  TH  ITERATION'/5X,'*****  THE ABOVE POINT IS AN
     &IMPROVED POINT  *****'/)
  145 FORMAT (/1X,I5,'  TH  ITERATION'/5X,'*****  THE ABOVE POINT IS NOT
     & AN IMPROVED POINT  *****'/)
  150 FORMAT (5X,'*****  AND ALSO THE CURRENT OPTIMAL POINT  *****'/)
  155 FORMAT (5X,'*****  BUT NOT THE CURRENT OPTIMAL POINT  *****'/)
  160 FORMAT (1H1)
  165 FORMAT (/5X,'*****   OPTIMAL SOLUTION   *****'///)
  170 FORMAT (/20X,'X(',I1,') = ',F14.6)
  180 FORMAT (/20X,'Y(',I1,') = ',F14.6)
  175 FORMAT (/70X,'X(',I1,') = ',F14.6)
  185 FORMAT (/70X,'Y(',I1,') = ',F14.6)
  200 FORMAT (/1X,I5,'  TH  ITERATION'/5X,'*****  THE ABOVE POINT IS THE
     & CHECKING CENTROID  *****'/)
 1000 FORMAT (/45X,'*****   LIST OF INPUT DATA   *****'///)
C
      STOP 5555
      END
      SUBROUTINE FSOPT(X,ICUT,Y)
```

```
C
C*****TH[illegible]SU[illegible]UTI[illegible]LOC[illegible]S [illegible]SI[illegible]PO[illegible]S
C
      DIMENSION X(30),Y(20)
      COMMON N,M,NR
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      ICUT=2
      DO 1 I=1,N
      IF (X(I).LT.XLP(I)) GO TO 3
      IF (X(I).GT.XUP(I)) GO TO 3
    1 CONTINUE
      ICUT=ICUT-1
C
      CALL SIMUL(X,Y,NR)
      REWIND 10
      DO 10 J=2,NR
      IF (Y(J).LT.YL) GO TO 3
      IF (Y(J).GT.YU) GO TO 3
   10 CONTINUE
      ICUT=ICUT-1
C
    3 RETURN
      END
C
      SUBROUTINE WORST(K,ZMIN,Z1)
C
C*****THIS SUBROUTINE INDICATES THE CURRENT WORST POINT
C
      DIMENSION Z1(20)
      COMMON N,M,NR
      K=1
      ZMIN=Z1(1)
      DO 1 J=2,M
      IF (Z1(J).GE.ZMIN) GO TO 1
      K=J
      ZMIN=Z1(K)
    1 CONTINUE
      RETURN
      END
C
      SUBROUTINE ZOPT(KP,ZMAX,Z1)
C
C*****THIS SUBROUTINE INDICATES THE CURRENT BEST POINT
C
      DIMENSION Z1(20)
      COMMON N,M,NR
      KP=1
      ZMAX=Z1(1)
      DO 1 J=2,M
      IF (Z1(J).LE.ZMAX) GO TO 1
      KP=J
      ZMAX=Z1(KP)
    1 CONTINUE
      RETURN
      END
C
      SUBROUTINE SIMUL(XS,Y,IIP1)
      DIMENSION XS(10),Y(20)
```

APPENDIX B

PROGRAM LISTING

FOR A

SEQUENTIAL FIRST-ORDER RESPONSE SURFACE METHOD

```
C
      DIMENSION XBAR(30),STD(30),RY(30),
     &          BV(30),SB(30),TV(30),RX(30,30),RT(465),ANS(10),
     &          XP(10),XC(10)
      COMMON X(40,30),XV(1200),XL(20),XU(20),YL(10),YU(10),
     &       Y(10),GRAD(20),N,MVTEX,AVTEX,NV,M,MOB,NR,STEP,RAD,
     &       RRRA,CONF,GOLD,ISTOP,NTV
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE2/ISEED1,ISEED2
      COMMON/BLEE3/XL(10),XU(10,YL,YU
      COMMON/BL4/JACT(30)
      COMMON/BL5/OV(30),ISAVE(30)
      COMMON/BLEE8/ITER
      WRITE (6,2000)
      READ (9,*) MDSIM
      READ (9,*) N,NR,MCP
      READ (9,*) (XL(J),J=1,N), (XU(J),J=1,N)
      READ (9,*) (XLP(I),I=1,N), (XUP(I),I=1,N)
      READ (9,*) (DB(J),J=1,9)
      READ (9,*) YL,YU
      READ (9,*) STEP,RAD,RRRA
      READ (9,*) CONF
      READ (9,*) IX
      READ (9,*) ISEED1,ISEED2
C
      WRITE (12,*) MDSIM
      WRITE (12,*) N,NR,MCP
      WRITE (12,*) (XL(J),J=1,N), (XU(J),J=1,N)
      WRITE (12,*) (XLP(I),I=1,N), (XUP(I),I=1,N)
      WRITE (12,*) (DB(J),J=1,9)
      WRITE (12,*) YL,YU
      WRITE (12,*) STEP,RAD,RRRA
      WRITE (12,*) CONF
      WRITE (12,*) IX
      WRITE (12,*) ISEED1,ISEED2
C
      GOLD=(-1.+SQRT(5.))/2.
      ITER=0
      GO TO (10,20), MODEV
   10 WRITE (6,2015)
      MVTEX=N+1
      GO TO 30
C
   20 WRITE (6,2025)
      MVTEX=2**N
C
   30 AVTEX=MVTEX
      M=MVTEX+1
      MP1=M+1
      MOB=MVTEX+MCP
      NV=N+1
      NTV=N+NR
      WRITE (6,2035)
      WRITE (6,2040) (YU(J),J=2,NR)
      DO 45 J=1,NTV
      ISAVE(J)=J
   45 CONTINUE
   50 DO 60 J=1,N
      CALL RANDU(IX,IY,RJMN)
```

```
      IX=IY
      DA=XU(J)-XL(J)
      X(M,J)=XL(J)+RUMN*DA
      XC(J)=X(M,J)
   60 CONTINUE
C
      CALL FSUPT(IOUT,JACT,XC)
      IF (IOUT.EQ.1) GO TO 50
      DO 80 K=1,NR
      J=N+K
      X(M,J)=Y(K)
      XC(J)=Y(K)
   80 CONTINUE
 1000 IF (MCP.EQ.1) GO TO 1050
      DO 90 I=MP1,MOB
      IF (MDSIM.EQ.1) GO TO 83
      CALL SIMUL(XC,Y)
   83 DO 85 K=1,NR
      J=N+K
      X(I,J)=Y(K)
   85 CONTINUE
   90 CONTINUE
      GO TO (1050,1060), MDSIM
 1050 WRITE (6,2050) ITER
      GO TO 95
 1060 WRITE (6,2060) ITER,MCP
   95 GO TO (100,150), MODEV
C
  100 CALL SIMPLX(XC)
      GO TO 1100
  150 CALL FACTOL(XC)
 1100 WRITE (6,2100) ITER,MVTEX
      DO 350 J=1,NV
      DO 350 I=1,MOB
      K=MOB*(J-1)+I
      XV(K)=X(I,J)
  350 CONTINUE
      CALL CORRE(MOB,NV,1,XV,XBAR,STD,RX,RT,BV,DV,TV)
      CALL ORDER(NV,RT,NV,N,ISAVE,RX,RY)
      CALL MINV(RX,N,DET,DV,TV)
      WRITE (6,320) ((X(I,J),J=1,NTV),I=1,MOB)
  320 FORMAT(/(5X,4F14.6))
      CALL MULTR(MOB,N,XBAR,STD,BV,RX,RY,ISAVE,DV,SB,TV,ANS)
      WRITE (6,2150) ITER,(XC(J),J=1,N), ANS(1), (DV(J),J=1,N)
      WRITE (6,2200) (DV(J),J=1,N)
      IF(IOUT.EQ.0) GO TO 380
      CALL PROJCT(JACT)
      WRITE (6,2300) (DV(J),J=1,N)
  380 SUMSQ=0.
      DO 400 J=1,N
      SUMSQ=SUMSQ+DV(J)*DV(J)
  400 CONTINUE
      DO 450 J=1,N
      GRAD(J)=DV(J)/SQRT(SUMSQ)
  450 CONTINUE
      CALL GOLDEN(IOUT,JACT,XC)
      IF (ISTOP.EQ.1) GO TO 500
      ITER=ITER+1
      GO TO 1000
```

```
  500 WRITE (6,3000) (XC(J),J=1,N)
      WRITE (6,3050) XC(NV)
      WRITE (6,3100) STEP
 2000 FORMAT (///5X,'*****  THIS TANKOUEL OPTIMIZATION USES :'///
     &   15X,'FIRST ORDER REGRESSION FOR THE LOCAL RESPONSE SURVEY')
 2015 FORMAT (/15X,'SIMPLEX DESIGN FOR VERTICES')
 2025 FORMAT (/15X,'FACTORIAL DESIGN FOR VERTICES')
 2035 FORMAT (/15X,'LINE SEARCH WITH GOLDEN SECTION')
 2040 FORMAT (/10X,'ROUNDS CARRIED BY BLUE IS',F7.1/)
 2050 FORMAT (/5X,I5,'  TH  ITERATION'/10X,
     &        '*****  THE ABOVE POINT IS THE CENTER POINT  *****'/)
 2060 FORMAT (/5X,I5,'  TH  ITERATION'/10X,
     &        '*****  THE ABOVE ',I4,2X,'POINTS ARE THE CENTER POINTS'/)
 2100 FORMAT (/5X,I5,'  TH  ITERATION'/10X,
     &        '*****  THE ABOVE ',I4,2X,'POINTS ARE THE VERTICES'/)
 2150 FORMAT (/5X,I5,'  TH  ITERATION'/10X,'THE LOCAL RESPONSE SURFACE A
     &BOUT ( X(1) =',F10.5,',  X(2) =',F10.5,' )  IS'//10X,F10.5,
     &'  +',F10.5,' X(1) +',F10.5,' X(2)  =  Y(1)')
 2200 FORMAT (/10X,'THE LOCAL STEEPEST ASCENT DIRECTION IS'//20X,
     &        '(  ',F10.5,'  ,  ',F10.5,'  )'/)
 2300 FORMAT (/10X,'THE LOCAL PROJECTION DIRECTION IS'//20X,
     &        '(  ',F10.5,'  ,  ',F10.5,'  )'/)
 3000 FORMAT (////5X,'OPTIMAL X(1) =',F10.5//5X,'OPTIMAL X(2) =',F10.5)
 3050 FORMAT (//5X,'OPTIMAL Y(1) =',F10.5)
 3100 FORMAT (///5X,'CONFIDENT RADIUS =',F10.5)
      STOP
      END
C
      SUBROUTINE SIMPLX(XC)
      DIMENSION XP(10),XC(10)
      COMMON X(40,30),XV(1200),XL(20),XU(20),YL(10),YU(10),
     &       Y(10),GRAD(20),N,MVTEX,AVTEX,NV,M,MOB,NR,STEP,RAD,
     &       RRRA,CONF,GOLD,ISTOP,NTV
      ANG=2.*3.14159/AVTEX
      ANGLE=0.
      DO 300 I=1,MVTEX
      X(I,1)=XC(1)+RAD*COS(ANGLE)
      X(I,2)=XC(2)+RAD*SIN(ANGLE)
      DO 80 J=1,N
      XP(J)=X(I,J)
   80 CONTINUE
      CALL SIMUL(XP,Y)
      DO 100 K=1,NR
      J=N+K
      X(I,J)=Y(K)
      XP(J)=Y(K)
  100 CONTINUE
      ANGLE=ANGLE+ANG
  300 CONTINUE
      RAD=RAD*RRRA
      RETURN
      END
C
      SUBROUTINE FACTOL(XC)
      DIMENSION XP(10),XC(10)
      COMMON X(40,30),XV(1200),XL(20),XU(20),YL(10),YU(10),
     &       Y(10),GRAD(20),N,MVTEX,AVTEX,NV,M,MOB,NR,STEP,RAD,
     &       RRRA,CONF,GOLD,ISTOP,NTV
      K1=1
```

```
      K2=1
      DO 300 KK1=1,2
      DO 200 KK2=1,2
      I=2*(KK1-1)+KK2
      X(I,1)=XC(1)+RAD*K1
      X(I,2)=XC(2)+RAD*K2
      DO 30 J=1,N
      XP(J)=X(I,J)
   30 CONTINUE
      CALL SIMUL(XP,Y)
      DO 100 K=1,NR
      J=N+K
      X(I,J)=Y(K)
      XP(J)=Y(K)
  100 CONTINUE
      K2=-K2
  200 CONTINUE
      K1=-K1
  300 CONTINUE
      RAD=RAD*RRRA
      RETURN
      END
C
      SUBROUTINE GOLDEN(ICUT,JACT,XC)
      DIMENSION XGL(10),XGU(10),XG1(10),XG2(10),XP(10),XC(10),KACT(20),
     &          JACT(30)
      COMMON X(40,30),XV(1200),XL(20),XU(20),YL(10),YU(10),
     &       Y(10),GRAD(20),N,MVTEX,AVTEX,NV,M,MOB,NR,STEP,RAD,
     &       RRRA,CONF,GOLD,ISTOP,NTV
      ISTOP=0
      IND=1
      ICUT=0
      Y1MAX=XC(NV)
      DO 50 J=1,NTV
      XGL(J)=XC(J)
   50 CONTINUE
   80 TSTEP=STEP
  100 DO 150 J=1,N
      XP(J)=XC(J)+TSTEP*GRAD(J)
  150 CONTINUE
      CALL FSHPT(JOUT,JACT,XP)
      ICUT=ICUT+JOUT
      IF(JOUT.EQ.1) GO TO 850
      GO TO (280,240,200), IND
  200 DO 220 J=1,NTV
      XGL(J)=XG1(J)
  220 CONTINUE
  240 DO 260 J=1,NTV
      XG1(J)=XGU(J)
  260 CONTINUE
  280 DO 300 J=1,N
      XGU(J)=XP(J)
  300 CONTINUE
      DO 320 K=1,NR
      J=N+K
      XGU(J)=Y(K)
  320 CONTINUE
      IND=IND+1
      IF (Y(1).LE.Y1MAX) GO TO 400
```

```
      IF (IOUT.NE.0) GO TO 430
      Y1MAX=Y(1)
      STEP=STEP/GOLD
      TSTEP=TSTEP+STEP
      GO TO 100
C
  400 IF (IND.NE.2) GO TO 490
  430 DO 450 J=1,N
      XG1(J)=XGL(J)+STEP*GOLD*GOLD*GRAD(J)
  450 CONTINUE
      CALL SIMUL(XG1,Y)
      DO 480 K=1,NR
      J=N+K
      XG1(J)=Y(K)
  480 CONTINUE
      GO TO 500
  490 STEP=STEP/GOLD
  500 DO 550 J=1,N
      XG2(J)=XGL(J)+STEP*GOLD*GRAD(J)
  550 CONTINUE
      CALL SIMUL(XG2,Y)
      DO 580 K=1,NR
      J=N+K
      XG2(J)=Y(K)
  580 CONTINUE
C
  600 STEP=STEP*GOLD
      IF (XG1(NV).LT.XG2(NV)) GO TO 700
      DO 650 J=1,NTV
      XG2(J)=XG1(J)
      XG1(J)=XGL(J)+STEP*GOLD*GOLD*GRAD(J)
  650 CONTINUE
      CALL SIMUL(XG1,Y)
      DO 680 K=1,NR
      J=N+K
      XG1(J)=Y(K)
  680 CONTINUE
      GO TO 800
  700 DO 750 J=1,NTV
      XGL(J)=XG1(J)
      XG1(J)=XG2(J)
      XG2(J)=XGL(J)+STEP*GOLD*GRAD(J)
  750 CONTINUE
      CALL SIMUL(XG2,Y)
      DO 780 K=1,NR
      J=N+K
      XG2(J)=Y(K)
  780 CONTINUE
C
  800 IF (STEP.LT.CONF) GO TO 950
      GO TO 600
  850 STEP=STEP*GOLD
      IF (IND.NE.1) GO TO 900
      IF (STEP.LT.CONF) GO TO 1200
      GO TO 80
  900 IF (IND.NE.2) GO TO 490
      GO TO 430
  950 IF (XG1(NV).LT.XG2(NV)) GO TO 1050
      DO 1000 J=1,NTV
```

```
      XC(J)=XGI(J)
      X(M,J)=XC(J)
 1000 CONTINUE
      GO TO 1300
 1050 DO 1100 J=1,NTV
      XC(J)=XG2(J)
      X(M,J)=XC(J)
 1100 CONTINUE
      GO TO 1300
 1200 ISTOP=1
 1300 RETURN
      END
C
      SUBROUTINE FSBPT(IOUT,JACT,XT)
      DIMENSION XT(2),JACT(30)
      COMMON X(40,30),XV(1200),XL(20),XU(20),YL(10),YU(10),
     &       Y(10),GRAD(20),N,MVTEX,AVTEX,NV,M,MOB,NR,STEP,RAD,
     &       RRRA,CONF,GOLD,ISTOP,NTV
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      IOUT=0
      CALL SIMUL(XT,Y,NR)
      DO 50 J=1,NTV
      JACT(J)=0
   50 CONTINUE
      DO 160 J=1,N
      IF (XT(J).GE.XLP(J)) GO TO 120
      GO TO 140
  120 IF (XT(J).LE.XUP(J)) GO TO 160
  140 JACT(J)=1
      IOUT=1
  160 CONTINUE
      DO 260 L=2,NR
      J=N+L
      IF (Y(L).GE.YL) GO TO 220
      GO TO 240
  220 IF (Y(L).LE.YU) GO TO 260
  240 JACT(J)=1
      IOUT=1
  260 CONTINUE
      RETURN
      END
C
      SUBROUTINE PROJCT(JACT)
      DIMENSION G(4,4),GTRA(4,4),GPRD(4,4),GPRDI(4,4),GV(16),
     &          DVI(30),IVI(30),XBAR(30),STD(30),RX(30,30),RT(465),
     &          RY(30),BV(30),SB(30),ARS(10),JACT(30)
      COMMON X(40,30),XV(1200),XL(20),XU(20),YL(10),YU(10),
     &       Y(10),GRAD(20),N,MVTEX,AVTEX,NV,M,MOB,NR,STEP,RAD,
     &       RRRA,CONF,GOLD,ISTOP,NTV
      COMMON/BLS/DV(30),ISAVE(30)
      ICONS=0
      DO 100 J=1,N
      IF (JACT(J).EQ.0) GO TO 100
      ICONS=ICONS+1
      DO 50 K=1,N
      G(ICONS,K)=0.
   50 CONTINUE
      G(ICONS,J)=1.
```

```
100 CONTINUE
    DO 250 L=2,NR
    J=N+L
    IF (JACT(J).EQ.0) GO TO 250
    ICONS=ICONS+1
    DO 150 I=1,MOB
    K=MOB*N+I
    XV(K)=X(I,J)
150 CONTINUE
    CALL CORRE(MOB,NV,1,XV,XBAR,STD,RX,RT,BV,DV1,TV1)
    CALL ORDER(NV,RT,NV,N,ISAVE,RX,RY)
    CALL MINV(RX,N,DET,DV1,TV1)
    CALL MULTR(MOB,N,XBAR,STD,BV,RX,RY,ISAVE,DV1,SB,TV1,ANS)
    DO 200 K=1,N
    G(ICONS,K)=DV1(K)
200 CONTINUE
250 CONTINUE
    DO 500 J=1,N
    DO 500 I=1,ICONS
    K=ICONS*(J-1)+I
    GV(K)=G(I,J)
500 CONTINUE
    DO 300 I=1,ICONS
    WRITE (6,1300) (G(I,J),J=1,N)
300 CONTINUE
1300 FORMAT (5X,8F14.6)
    CALL GMTRA(GV,GTRA,ICONS,N)
    CALL GMPRD(GV,GTRA,GPRD,ICONS,N,ICONS)
    CALL MINV(GPRD,ICONS,DET,DV1,TV1)
    CALL GMPRD(GTRA,GPRD,GPRD1,N,ICONS,ICONS)
    CALL GMPRD(GPRD1,GV,GPRD,N,ICONS,N)
    CALL GMPRD(GPRD,DV,GPRD1,N,N,1)
    CALL GMSUB(DV,GPRD1,DV,N,1)
    RETURN
    END
C
    SUBROUTINE DATA(NV,DV)
    DIMENSION DV(1)
  1 FORMAT (12F6.0)
    READ (3,1) (DV(J),J=1,NV)
    WRITE (1) (DV(J),J=1,NV)
    RETURN
    END
    SUBROUTINE SIMUL(XS,Y)
C
C*****THIS IS THE AMSAA TANK DUEL SIMULATION MODEL
C
    DIMENSION XS(30), Y(20)
    DIMENSION TMDB(45),TMDR(45),SDB(45),SDR(45),SKB(45),SKR(45)
    COMMON/BL1/BPHI,BPHH,BPHS,BPHL,BKHI,BKHH,BKHS,BKHL,BS
    COMMON/BL2/TRI,TRS,RPHI,RPHH,RPHS,RPHL,RKHI,RKHH,RKHS,RKHL,RS
    COMMON/BL3/ITER
    REAL KB(45),KR(45),M(40,40),N(40,40),NDF,NODEC
    INTEGER RANGE
    DATA SIGMA,BLUE,RED,TCUT,LS,TFB,TTB,ID1,ID2,ID3,RANGE,
   1RFLD,IRR,ITR,ID4,ID5,ID6,RCLR/.5,4HBLUE,3HRED,120.0,0,
   2.35,7.,3HBLU,3HBLU,3HBLU,1000,.35,1.17,7.,3HRED,3HRED,
   33HRED,.825/
    TB1=XS(1)
```

```
      END
      SUBROUTINE CONLOG(XI,SIGX,ETA,SIGY,ZETA,SIGZ)
C
C*****THIS SUBROUTINE COMPUTES THE MEDIAN AND STANDARD DEVIATION FOR
C*****CONVOLUTION
C
      XBAR=XI*EXP(.5*SIGX*SIGX)
      YBAR=ETA*EXP(.5*SIGY*SIGY)
      SSX=XBAR*XBAR*(EXP(SIGX*SIGX)-1.0)
      SSY=YBAR*YBAR*(EXP(SIGY*SIGY)-1.0)
      ESIGZ=1.0+((SSX+SSY)/((XBAR+YBAR)**2))
      ZETA=(XBAR+YBAR)/SQRT(ESIGZ)
      SIGZ=SQRT(ALOG(ESIGZ))
      RETURN
      END
      SUBROUTINE KASFT(K,SK,JOUT,KHI,KHH,KHS,KHL,PHI,PHH,PHS,PHL,S,R)
C
C*****THIS SUBROUTINE COMPUTES THE KILL AND SURVIVAL FOR THE TWO TANKS.
C
      DIMENSION SK(45)
      REAL K(45),KHI,KHH,KHS,KHL,L
      DO 100 I=2,45
      K(I)=0.0
100   SK(I)=0.0
      K(1)=PHI*KHI*R
      SK(1)=1.0-K(1)
      L=1.0-S
      X2=PHI*(1.0-KHI*R)
      X3=(1.0-PHI)*S
      X4=(1.0-PHI)*L
      A12=PHH*KHH*R
      A13=PHS*KHS*R
      A14=PHL*KHL*R
      A22=PHH*(1.0-KHH*R)
      A23=PHS*(1.0-KHS*R)
      A24=PHL*(1.0-KHL*R)
      A32=(1.0-PHH)*S
      A33=(1.0-PHS)*S
      A34=(1.0-PHL)*S
      A42=(1.0-PHH)*L
      A43=(1.0-PHS)*L
      A44=(1.0-PHL)*L
      DO 130 I=2,45
      K(I)=A12*X2+A13*X3+A14*X4
      X3P=A32*X2+A33*X3+A34*X4
      X2P=A22*X2+A23*X3+A24*X4
      X4P=A42*X2+A43*X3+A44*X4
      X2=X2P
      X3=X3P
      X4=X4P
      JOUT=I
      SK(I)=SK(I-1)-K(I)
      IF (I.LT.11) GOTO 130
      IF (SK(I-5).LT..0005) GOTO 135
130   CONTINUE
135   RETURN
      END
      FUNCTION PABAT(I,TA,TB,SA,SB)
C
```

APPENDIX C

PROGRAM LISTING

FOR A

SECOND-ORDER RESPONSE SURFACE METHOD

BASED ON A CENTRAL COMPOSITE DESIGN

```
C
C
C*****   THIS OPTIMIZATION TECHNIQUE USES :
C*****       1.  CENTRAL COMPOSITE DESIGN FOR THE VERTICES
C*****       2.  SECOND ORDER REGRESSION FOR THE RESPONSE SURFACE SURVEY
C*****       3.  BOX'S COMPLEX SEARCH ON THE ESTIMATED RESPONSE SURFACES
C*****            N      :   NO. OF INDEPENDENT VARIABLE TO BE OPTIMIZED
C*****            NR     :   NO. OF DEPENDENT VARIABLE (RESPONSES)
C*****            MCP    :   WEIGHT OF CENTER POINT IN REGRESSION
C*****            M      :   NO. OF EXPERIMENTS FOR EACH REGRESSION
C*****            MOB    :   NO. OF OBSERVATIONS IN EACH REGRESSION
C
      DIMENSION X(10,30),XP(30),C(10),Y(20),Y1(10),Y2(10,10),DA(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MDSIM,NP
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE2/ISEED1,ISEED2
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BL3/RAD(10),XC(10)
      COMMON/BLEE8/ITER
      COMMON/BL15/KOUPK,DUPK
C
      READ (9,*) MDSIM
      READ (9,*) N,MCP,MPBC
      READ (9,*) (XL(J),J=1,N),(XU(J),J=1,N)
      READ (9,*) (XLP(J),J=1,N),(XUP(J),J=1,N)
      READ (9,*) YL,YU
      READ (9,*) KOUPK
      READ (9,*) IX
      READ (9,*) ISEED1,ISEED2
C
      WRITE (12,1050)
      WRITE (12,*) MDSIM
      WRITE (12,*) N,MCP,MPBC
      WRITE (12,*) (XL(J),J=1,N),(XU(J),J=1,N)
      WRITE (12,*) (XLP(J),J=1,N),(XUP(J),J=1,N)
      WRITE (12,*) YL,YU
      WRITE (12,*) KOUPK
      WRITE (12,*) IX
      WRITE (12,*) ISEED1,ISEED2
C
      DUPK=KOUPK
      AN=N
      NP=N+MPBC
      NPM1=NP-1
      ANPM1=NPM1
      ITER=0
      ITER2=0
C
      DO 100 J=1,N
      DA(J)=XUP(J)-XLP(J)
      RAD(J)=DA(J)/2.
      XC(J)=(XUP(J)+XLP(J))/2.
  100 CONTINUE
C
      WRITE (12,5000)
      CALL CENCOM
      CALL REGRES
C
      MAX=1
```

```
      DO 220 I=1,NP
  120 DO 140 J=1,N
      CALL RANDU(IX,IY,RUMN)
      IX=IY
      XP(J)=XLP(J)+RUMN*DA(J)
      X(I,J)=XP(J)
  140 CONTINUE
      CALL SECIZE(XP)
      CALL FSBPT(XP,Y,IOUT)
      IF (IOUT.EQ.2) GO TO 120
      ITER2=ITER2+1
      WRITE (12,1450) ITER2
      CALL WRITE(XP,Y,N,NR,IOUT)
      IF (IOUT.NE.0) GO TO 120
      DO 200 J=1,NR
      YP(I,J)=Y(J)
  200 CONTINUE
      Y1(I)=Y(1)
      IF (Y(1).GT.Y1(MAX)) MAX=I
  220 CONTINUE
      WRITE (12,1230) NP
C
  300 CALL WORST(Y1,MIN)
      DO 340 J=1,N
      SUM=-X(MIN,J)
      DO 320 I=1,NP
      SUM=SUM+X(I,J)
  320 CONTINUE
      C(J)=SUM/ANPM1
  340 CONTINUE
C
      R=2.
  400 R=R*0.5
      IF (R.LT.1.E-2) GO TO 600
      DO 420 J=1,N
      XP(J)=C(J)+R*(C(J)-X(MIN,J))
  420 CONTINUE
      CALL SECIZE(XP)
      CALL FSBPT(XP,Y,IOUT)
      IF (IOUT.EQ.2) GO TO 400
      ITER2=ITER2+1
      WRITE (12,1450) ITER2
      CALL WRITE(XP,Y,N,NR,IOUT)
      IF (IOUT.NE.0) GO TO 400
      IF (Y(1).LE.Y1(MIN)) GO TO 400
      DO 440 J=1,NR
      YP(MIN,J)=Y(J)
  440 CONTINUE
      DO 500 J=1,N
      X(MIN,J)=XP(J)
  500 CONTINUE
      Y1(MIN)=Y(1)
      IF (Y1(MIN).GT.Y1(MAX)) MAX=MIN
      GO TO 300
C
  600 WRITE (12,1600)
      DO 620 J=1,N
      WRITE (12,1160) J,X(MAX,J)
  620 CONTINUE
```

```
      DO 640 J=1,NR
      WRITE (12,1180) J,YP(MAX,J)
  640 CONTINUE
C
      DO 700 J=1,N
      XP(J)=X(MAX,J)
  700 CONTINUE
      CALL AVGSIM(XP,Y,N,NR)
      WRITE (12,1600)
      CALL WRITE(XP,Y,N,NR,0)
C
 1050 FORMAT (1H1/35X,'*****    LIST OF INPUT DATA    *****'//)
 1100 FORMAT (30X,'X(',I1,') = ',F14.6)
 1180 FORMAT (30X,'Y(',I1,') = ',F14.6)
 1230 FORMAT (//5X,'THE ABOVE ',I2,' POINTS ARE THE INITIAL POINTS')
 1450 FORMAT (//1X,I3,' TH ITERATION'//)
 1600 FORMAT (////5X,'OPTIMAL SOLUTION'//)
 5000 FORMAT (1H1)
      STOP
      END
C
       SUBROUTINE CENCOM
C
C*****  THIS SUBROUTINE GENERATES THE EXPERIMENT POINTS FOR THE
C*****  CENTRAL COMPOSITE DESIGN
C
      DIMENSION Y(20),ED(10),XP(30)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MDSIM,NP
      COMMON/BL5/RAD(10),XC(10)
      COMMON/BL7/XE(1000),YE(1000)
      COMMON/BL9/EX(1000),EY(1000)
C
      SQRTN=SQRT(AN)
      MFAC=2**N
      MAXIS=2*N
      MVTEX=MFAC+MAXIS
      M=MVTEX+1
      MOB=MVTEX+MCP
      NM1=N-1
      N2D=N*(N+3)/2
      N2DP1=N2D+1
C
      DO 50 J=1,N
      ED(J)=RAD(J)/SQRTN
   50 CONTINUE
C
      KX=0
      KY=0
      DO 160 K=1,MFAC
      KA=K
      DO 120 J=1,N
      KB=MOD(KA,2)
      KA=KA/2
      XP(J)=XC(J)+ED(J)*(-1)**KB
  120 CONTINUE
      CALL SECIZE(XP)
      CALL AVGSIM(XP,Y,N,NR)
      CALL RECORD(XP,Y,KX,KY)
  160 CONTINUE
```

```
C
      DO 280 I=1,N
      DO 260 IA=1,2
      DO 220 J=1,N
      XP(J)=XC(J)
      IF (I.EQ.J) XP(J)=XC(J)+RAD(J)*(-1)**IA
  220 CONTINUE
      CALL SECIZE(XP)
      CALL AVGSIM(XP,Y,N,NR)
      CALL RECORD(XP,Y,KX,KY)
  260 CONTINUE
  280 CONTINUE
C
      CALL SECIZE(XC)
      CALL AVGSIM(XC,Y,N,NR)
      CALL RECORD(XC,Y,KX,KY)
C
      IF (MCP.EQ.1) GO TO 500
      DO 400 I=2,MCP
      IF(MDSIM.EQ.1) GO TO 300
      CALL AVGSIM(XC,Y,N,NR)
  300 CALL RECORD(XC,Y,KX,KY)
  400 CONTINUE
C
      WRITE (12,1450)
  500 DO 600 I=1,MOB
      KX1=(I-1)*N2D+1
      KX2=I*N2D
      KY1=(I-1)*NR+1
      KY2=I*NR
      WRITE (12,1600) I,(EX(J),J=KX1,KX2),(EY(J),J=KY1,KY2)
  600 CONTINUE
C
      CALL GMTRA(EX,XE,N2D,MOB)
      CALL GMTRA(EY,YE,NR,MOB)
C
 1450 FORMAT (////50X,'REGRESSION TABLEAU '//12X,'X1',12X,'X2',11X,
     &'X1SQ',10X,'X2SQ',10X,'X1X2',11X,'Y1',12X,'Y2',12X,'Y3',12X,'Y4'/
     &12X,'Y5',12X,'Y6')
 1600 FORMAT(/1X,I3,3X,9(F12.6,2X)/7X,9(F12.6,2X)/)
      RETURN
      END
C
      SUBROUTINE SECIZE(XS)
C
C*****   THIS SUBROUTINE TRANSFORMS 1ST ORDER INNDEPENDENT VARIABLES
C*****   INTO 2ND ORDER INDEPENDENT VARIABLES FOR
C*****   2ND ORDER REGRESSION
C
      DIMENSION XS(30)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MDSIM,NP
      K=N
      DO 100 J=1,N
      K=K+1
      XS(K)=XS(J)*XS(J)
  100 CONTINUE
C
      DO 300 I=1,NM1
      IP1=I+1
```

```
      DO 200 J=IP1,N
      K=K+1
      XS(K)=XS(I)*XS(J)
  200 CONTINUE
  300 CONTINUE
      RETURN
      END
C
      SUBROUTINE REGRES
C
C*****  THIS SUBROUTINE CALCULATES THE REGRESSION COEFFICIENTS OF THE
C*****  RESPONSE SURFACES
C
      DIMENSION XBAR(30),STD(30),RX(900),RT(465),BV(30),DV(30),TV(30),
     E           ISAVE(30),RY(30),SB(30),ANS(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MOSIM,NP
      COMMON/BL6/B(10,30)
      COMMON/BL7/XE(1000),YE(1000)
C
      NXE=MOB*N2D
C
      DO 50 J=1,N2DP1
      ISAVE(J)=J
   50 CONTINUE
C
C
      WRITE (12,1150)
      DO 300 J=1,NR
      KXE=NXE
      KYE=MOB*(J-1)
      DO 100 I=1,MOB
      KXE=KXE+1
      KYE=KYE+1
      XE(KXE)=YE(KYE)
  100 CONTINUE
C
      CALL CORRE(MOB,N2DP1,1,XE,XBAR,STD,RX,RT,BV,DV,TV)
      CALL ORDER(N2DP1,RT,N2DP1,N2D,ISAVE,RX,RY)
      CALL MINV(RX,N2D,DET,DV,TV)
      CALL MULTR(MOB,N2D,XBAR,STD,BV,RX,RY,ISAVE,DV,SB,TV,ANS)
      WRITE (12,1160)(DV(L),L=1,N2D),ANS(1),J
      WRITE (12,1170) ANS(10)
C
      DO 200 JCOEF=1,N2D
      B(J,JCOEF)=DV(JCOEF)
  200 CONTINUE
      B(J,N2DP1)=ANS(1)
  300 CONTINUE
C
 1150 FORMAT (////1X,'THE ESTIMATED RESPONSE SURFACES ARE :')
 1160 FORMAT (/5X,F12.6,' X(1) + ',F12.6,' X(2) + ',F12.6,
     &' X(1)SQ + ',F12.6,' X(2)SQ + ',F12.6,' X(1)*X(2) + ',
     &F12.6,' = Y(',I1,')')
 1170 FORMAT (//5X,'F VALUE   =   ',F14.6)
      RETURN
      END
C
      SUBROUTINE FSBPT(XT,Y,IOUT)
C
```

```
C*****  THIS SUBROUTINE CALCULATES THE RESPONSES  AND
C*****  DETECTS IF THEY ARE FEASIBLE
C
        DIMENSION XT(30),Y(20)
        COMMON N,AN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MDSIM,NP
        COMMON/BLEE3/XL(10),XU(10),YL,YU
        COMMON/BLEE1/XLP(10),XUP(10)
        COMMON/BL6/B(10,30)
        IOUT=2
C
        DO 100 J=1,N
        IF (XT(J).LT.XLP(J)) GO TO 500
        IF (XT(J).GT.XUP(J)) GO TO 500
  100   CONTINUE
        IOUT=1
C
        DO 300 I=2,NR
        Y(I)=B(I,N2DP1)
        DO 200 J=1,N2D
        Y(I)=Y(I)+B(I,J)*XT(J)
  200   CONTINUE
        IF (Y(I).LT.YL) GO TO 500
        IF (Y(I).GT.YU) GO TO 500
  300   CONTINUE
        Y(1)=B(1,N2DP1)
        DO 350 J=1,N2D
        Y(1)=Y(1)+B(1,J)*XT(J)
  350   CONTINUE
        IOUT=0
C
  500   RETURN
        END
C
        SUBROUTINE WORST(YT,MIN)
C
C*****  THIS SUBROUTINE FINDS OUT THE WPORST POINT IN THE
C*****  POLYGON
C
        DIMENSION YT(30)
        COMMON N,AN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MDSIM,NP
C
        MIN=1
        YMIN=YT(1)
        DO 50 J=2,NP
        IF (YT(J).GE.YMIN) GO TO 50
        MIN=J
        YMIN=YT(J)
   50   CONTINUE
        RETURN
        END
C
        SUBROUTINE DATA(NV,DV)
C
C*****  THIS SUBROUTINE IS CALLED BY THE SSP SUBROUTINE 'CORRE'
C*****  TO READ IN THE EXPERIMENT DATA NEEDED FOR REGRESSION
C
        DIMENSION DV(1)
    1   FORMAT (12F6.0)
        READ (3,1) (DV(J),J=1,NV)
```

```
      WRITE (1) (DV(J),J=1,NV)
      RETURN
      END
C
      SUBROUTINE WRITE(XW,YW,N,NR,ICODE)
      DIMENSION XW(30),YW(20)
C
      IF (ICODE.NE.0) GO TO 500
C
      DO 100 J=1,N
      WRITE (12,1100) J,XW(J)
  100 CONTINUE
      DO 200 J=1,NR
      WRITE (12,1200) J,YW(J)
  200 CONTINUE
      GO TO 1000
C
  500 DO 600 J=1,N
      WRITE (12,1600) J,XW(J)
  600 CONTINUE
      DO 700 J=1,NR
      WRITE (12,1700) J,YW(J)
  700 CONTINUE
C
 1100 FORMAT (30X,'X(',I1,') = ',F14.6)
 1200 FORMAT (30X,'Y(',I1,') = ',F14.6)
 1600 FORMAT (80X,'X(',I1,') = ',F14.6)
 1700 FORMAT (80X,'Y(',I1,') = ',F14.6)
 1000 RETURN
      END
C
      SUBROUTINE RECORD(XP,Y,KX,KY)
      DIMENSION XP(30),Y(20)
      COMMON N,NN,NM1,N2D,N2DP1,NR,M,MCP,MOB,MDSIM,NP
      COMMON/BL9/EX(1000),EY(1000)
C
      DO 100 J=1,N2D
      KX=KX+1
      EX(KX)=XP(J)
  100 CONTINUE
      DO 200 J=1,NR
      KY=KY+1
      EY(KY)=Y(J)
  200 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE AVGSIM(XA,Y,N,NR)
      DIMENSION XA(10),Y(20),YY(20)
      COMMON/BL15/KDUPK,DUPK
C
      I=1
      IF (KDUPK.NE.1) WRITE (12,1120) I
      CALL SIMUL(XA,Y,NP,1)
      REWIND 10
      IF (KDUPK.EQ.1) GO TO 1000
C
      DO 300 I=2,KDUPK
```

```
      WRITE (12,1120) I
      CALL SIMUL(XA,YY,NR,I)
      REWIND 10
      DO 200 J=1,NR
      Y(J)=Y(J)+YY(J)
  200 CONTINUE
  300 CONTINUE
      DO 400 J=1,NR
      Y(J)=Y(J)/DUPK
  400 CONTINUE
      WRITE (12,1400)
 1000 CALL WRITE(XA,Y,N,NR,1)
C
 1120 FORMAT (/1X,'** SIMUL. **',I3/)
 1400 FORMAT (//1X,'*****  AVERAGE RESULT   *****'//)
      RETURN
      END
C
C
      SUBROUTINE SIMUL(XS,Y,IIP1,KTER)
      DIMENSION XS(10),Y(20)
      COMMON /SUMMRY/ TKACT,TKDET,TITSNK,TITCUT,TNEUAT,TNESUC,TNUEAL,
     1   TLOST,TWAST,RUN
      COMMON /IPRNTX/ IPRINT(25)
     1        /ULIMIT/UPOS,UNEG
     2        /BLCKH/ENTIM(50),BTIME(300),CNTIM(300)
     3        /BLCKJ/MINTYP(300),MINCNT(300)
     4        /BLCKM/NSHTYP(50),TTIME,NHUNT,NSHIP,NTRAFF,JGROUP,NMINE
     5        /BLCKO/DFT(10),DPATH(300),DEPTH(300),TOP(10),DMGSW,THRSW
      COMMON/BLCKT/RESTIM(8,10),STRTIM(8,10),IOTRAN(10),KCURR(50),
     1             NPER(10),PLATIM
     2        /BLCKUU/PMINX(300),PMINY(300)
     3        /BLCKXX/TEMPCR,CASTIM,TIMSAV,TIMMAX
      COMMON/BLCKWW/DONTIM(300),NORMTH,CW2,MDXMAX(45),NAVMAX,NGP1,
     1                 INFTYP,KONTYP
     2        /BLCKZZ/MINLCK(300)
      COMMON/SAVE/PTORE(100,4),QTORE(100,4),TIIN,PQORE(100,4),QPORE(100,
     14)
      COMMON/DAMAGE/SDMG(5,10),NRUN,KSMPLT(10),FREQ(16,4),PTOTA(4),QTOTA
     1(4)
      COMMON/FLEG/JFLEG
      COMMON/BLEE8/ITER
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE2/ISEED1,ISEED2
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BLEE16/LSEED
      REAL*8 SEED
C
C INITIALIZING FOR TOTAL AVERAGES IN OUTPUT
C
      TKACT=0
      TKDET=0
      TITSN=0
      TITCUT=0
      TNEUAT=0
      TNESUC=0
      TNEUAL=0
      TLOST=0
      TWAST=0
```

APPENDIX D

PROGRAM LISTING

FOR A

SECOND-ORDER RESPONSE SURFACE METHOD

BASED ON BOX'S COMPLEX SEARCH

```
C
C*****  THIS PROGRAM USES THE FOLLOWING SCHEMES:
C        1. USE BOX'S COMPLEX SEARCH TO FIND THE OPTIMAL SOLUTION
C        2. CHECK IF THERE IS ANY SUBREGION WHICH HAS EXPERIMENT(S) LESS
C           THAN REQUIRED. DO COMPENSATE EXPERIMENT(S) IN IT.
C        3. FROM THE DATA RECORDED IN STEP 1 & 2 , CALCULATE THE
C           REGRESSION FUNCTIONS
C        4. USE THE REGRESSION FUNCTIONS TO FIND THE OPTIMAL SOLUTION WITH
C           BOX'S COMPLES SEARCH
C***  NOTATIONS  ***
C                 N :  NO. OF INPUT VARIABLES
C                NR :  NO. OF RESPONSES
C                NP :  NO. OF VERTICES OF THE POLYGON IN THE COMPLEX SEARCH
C             SMALL :  TERMINATING CRETERIA
C             MIEXR :  MIN. NO. OF EXPERIMENTS IN EACH SUBREGION
C          JSECT(J) :  NO. OF SUBSECTIONS THE (J)TH VARIABLE IS DEVIDED INTO
C             MAEXP :  MAX. NO. OF EXPERIMENTS IN THE FIRST COMPLEX SEARCH
C
      DIMENSION DA(10),XLR(10),XUR(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE2/ISEED1,ISEED2
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BL5/ICHK
      COMMON/BLEE8/ITER
      COMMON/BL10/MAEXP,SMALL(2)
      COMMON/BL11/JSECT(10),JAP(10)
      COMMON/BL12/KEXRG(1000)
      COMMON/BL13/IFLGN,MIEXR
      COMMON/BL14/ITER2
      COMMON/BL15/KDUPK,DUPK
C
      READ (9,*) N,MPBC
      READ (9,*) (XL(J),J=1,N),(XU(J),J=1,N)
      READ (9,*) (XLP(J),J=1,N),(XUP(J),J=1,N)
      READ (9,*) YL,YU
      READ (9,*) MAEXP,MAEXR,SMALL(1),SMALL(2)
      READ (9,*) (JSECT(J),J=1,N)
      READ (9,*) MIEXR
      READ (9,*) ICHK
      READ (9,*) IX
      READ (9,*) ISEED1,ISEED2
      READ (9,*) KDUPK
C
      WRITE (12,1050)
      WRITE (12,*) N,MPBC
      WRITE (12,*) (XL(J),J=1,N),(XU(J),J=1,N)
      WRITE (12,*) (XLP(J),J=1,N),(XUP(J),J=1,N)
      WRITE (12,*) YL,YU
      WRITE (12,*) MAEXP,MAEXR,SMALL(1),SMALL(2)
      WRITE (12,*) (JSECT(J),J=1,N)
      WRITE (12,*) MIEXR
      WRITE (12,*) ICHK
      WRITE (12,*) IX
      WRITE (12,*) ISEED1,ISEED2
      WRITE (12,*) KDUPK
C
      DUPK=KDUPK
      KX=0
```

```
      KY=0
C
      DO 100 J=1,N
      DA(J)=XUP(J)-XLP(J)
      SECT=JSECT(J)
      DAP(J)=DA(J)/SECT
  100 CONTINUE
C
      IREGN=1
      DO 300 J=1,N
      IREGN=IREGN*JSECT(J)-
  300 CONTINUE
C
      DO 500 K=1,IREGN
      KEXRG(K)=0
  500 CONTINUE
C
C
      AN=N
      NP=N+NPBC
      NPM1=NP-1
      ANP=NP
      ANPM1=NPM1
      N2D=N*(N+3)/2
C
      MEXP=0
      ITER=0
      ITER2=10000
      IBOX=1
      WRITE (12,5001)
      CALL BOXCOM(IBOX,IX,KX,KY,XLP,XUP,DA,Y1MAX)
C
      IF(MIEXR.EQ.0) GO TO 1000
      WRITE (12,1400)
      CALL COMPEN(KX,KY,ITRY,XLR,XUR,Y1MAX)
C
      IF (ITRY.EQ.0) GO TO 1000
      ITER2=30000
      MAEXP=MEXP+MAEXR
      WRITE (12,5001)
      CALL BOXCOM(IBOX,IX,KX,KY,XLR,XUR,DAP,Y1MAX)
C
 1000 WRITE (12,5001)
      CALL REGRES
C
      WRITE (12,5001)
      IBOX=2
      ITER2=20000
      CALL BOXCOM(IBOX,IX,KX,KY,XLP,XUP,DA,Y1MAX)
C
 1050 FORMAT (//45X,'*****   LIST OF INPUT DATA    *****'////)
 1400 FORMAT (1H1//45X,'*****    COMPENSATE EXPERIMENTS    *****'////)
 5001 FORMAT (1H1)
      STOP
      END
C
      SUBROUTINE BOXCOM(IBOX,IX,KX,KY,XL,XU,DA,Y1MAX)
      DIMENSION X(10,30),XP(30),C(10),Y(10),YP(10,10),XL(10),XU(10),DA(1
     &0),Y1(10)
```

```
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE2/ISEED1,ISEED2
      COMMON/BL5/ICHK
      COMMON/BLEE8/ITER
      COMMON/BL10/MAEXP,SMALL(2)
      COMMON/BL14/ITER2
C
      WRITE (12,1700)
      MAX=1
C
      DO 220 I=1,NP
  120 DO 140 J=1,N
      CALL RANDU(IX,IY,RNUM)
      IX=IY
      XP(J)=XL(J)+RNUM*DA(J)
      X(I,J)=XP(J)
  140 CONTINUE
      CALL SECIZE(XP)
      IF(IBOX.EQ.1) CALL FSBPT1(XP,Y,IOUT)
      IF(IBOX.EQ.2) CALL FSBPT2(XP,Y,IOUT)
      IF(IOUT.EQ.2) GO TO 120
      ITER2=ITER2+1
      WRITE (12,1450) ITER2
      CALL WRITE(XP,Y,N,NR,IOUT)
      IF(IBOX.EQ.2) GO TO 160
      CALL RECORD(XP,Y,KX,KY)
      MEXP=MEXP+1
      CALL REGION(XP)
  160 IF (IOUT.EQ.1) GO TO 120
      DO 200 J=1,NR
      YP(I,J)=Y(J)
  200 CONTINUE
      Y1(I)=Y(1)
      IF (Y(1).GT.Y1(MAX)) MAX=I
  220 CONTINUE
      WRITE (12,1230) NP
C
  300 CALL WORST(Y1,MIN)
      DO 340 J=1,N
      SUM=-X(MIN,J)
      DO 320 I=1,NP
      SUM=SUM+X(I,J)
  320 CONTINUE
      C(J)=SUM/ANPM1
  340 CONTINUE
C
      R=2.
  400 R=R*0.5
      DO 420 J=1,N
      XP(J)=C(J)+R*(C(J)-X(MIN,J))
  420 CONTINUE
      CALL SECIZE(XP)
      IF(IBOX.EQ.1) CALL FSBPT1(XP,Y,IOUT)
      IF(IBOX.EQ.2) CALL FSBPT2(XP,Y,IOUT)
      IF(IOUT.EQ.2) GO TO 400
      IF(IBOX.EQ.2) GO TO 430
      ITER2=ITER2+1
      WRITE (12,1450) ITER2
```

```
      CALL WRITE(XP,Y,N,NR,IOUT)
      CALL RECORD(XP,Y,KX,KY)
      MEXP=MEXP+1
      CALL REGION(XP)
      IF(MEXP.GE.MAEXP.AND.IOUT.EQ.1) GO TO 600
  430 IF (IOUT.EQ.1) GO TO 400
      IF (Y(1).LE.Y1(MIN)) GO TO 400
      DO 500 J=1,N
      X(MIN,J)=XP(J)
  500 CONTINUE
      DO 440 J=1,NR
      YP(MIN,J)=Y(J)
  440 CONTINUE
      Y1(MIN)=Y(1)
      IF (Y1(MIN).GT.Y1(MAX)) MAX=MIN
      IF(IBOX.EQ.1.AND.MEXP.GE.MAEXP) GO TO 600
C
      DO 510 I=1,NP
      A=Y1(MAX)-Y1(I)
      IF (A.GT.SMALL(IBOX)) GO TO 300
  510 CONTINUE
C
      IF (ICHK.EQ.0) GO TO 600
      DO 530 J=1,N
      SUM=X(1,J)
      DO 520 I=2,NP
      SUM=SUM+X(I,J)
  520 CONTINUE
      XP(J)=SUM/ANP
  530 CONTINUE
C
      CALL SECIZE(XP)
      IF(IBOX.EQ.1) CALL FSBPT1(XP,Y,IOUT)
      IF(IBOX.EQ.2) CALL FSBPT2(XP,Y,IOUT)
      IF (IBOX.EQ.2) GO TO 540
      CALL RECORD(XP,Y,KX,KY)
      MEXP=MEXP+1
      CALL REGION(XP)
  540 ITER2=ITER2+1
      WRITE (12,1450) ITER2
      WRITE (12,1500)
      CALL WRITE(XP,Y,N,NR,IOUT)
C
      DO 550 I=1,NP
      A=Y(1)-Y1(I)
      IF (A.GT.SMALL(IBOX)) GO TO 560
  550 CONTINUE
      GO TO 600
C
  560 CALL WORST(Y1,MIN)
      IF (Y(1).LE.Y1(MIN)) GO TO 600
      DO 570 J=1,N
      X(MIN,J)=XP(J)
  570 CONTINUE
      DO 580 J=1,NR
      YP(MIN,J)=Y(J)
  580 CONTINUE
      Y1(MIN)=Y(1)
      IF (Y1(MAX).GT.Y1(MIN)) GO TO 300
```

```
      MAX=MIN
      GO TO 300
C
  600 WRITE (12,1600)
      DO 620 J=1,N
      WRITE (12,1160) J,X(MAX,J)
  620 CONTINUE
      DO 640 J=1,NR
      WRITE (12,1180) J,YP(MAX,J)
  640 CONTINUE
C
      Y1MAX=Y1(MAX)
      IF (IBOX.EQ.1) RETURN
      DO 700 J=1,N
      XP(J)=X(MAX,J)
  700 CONTINUE
      CALL FSBPT1(XP,Y,IOUT)
      WRITE (12,1600)
      CALL WRITE(XP,Y,N,NR,IOUT)
C
 1160 FORMAT (30X,'X(',I1,') = ',F14.6)
 1180 FORMAT (30X,'Y(',I1,') = ',F14.6)
 1230 FORMAT (//5X,'THE ABOVE ',I2,' POINTS ARE THE INITIAL POINTS'//)
 1450 FORMAT (///1X,I5,' TH ITERATION'/)
 1500 FORMAT (7X,'**  CENTROID CHECKING POINT  **'/)
 1600 FORMAT (////5X,'OPTIMAL SOLUTION'//)
 1700 FORMAT (//27X,'*****   FEASIBLE POINT   *****',15X,
     &          '*****   INFEASIBLE POINT   *****')
C
 5000 FORMAT (//)
      RETURN
      END
C
C
      SUBROUTINE SECIZE(XS)
C
C*****  THIS SUBROUTINE TRANSFORMS 1ST ORDER INNDEPENDENT VARIABLES
C*****  INTO 2ND ORDER INDEPENDENT VARIABLES FOR
C*****  2ND ORDER REGRESSION
C
      DIMENSION XS(30)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      K=N
      DO 100 J=1,N
      K=K+1
      XS(K)=XS(J)*XS(J)
  100 CONTINUE
C
      DO 300 I=1,NM1
      IP1=I+1
      DO 200 J=IP1,N
      K=K+1
      XS(K)=XS(I)*XS(J)
  200 CONTINUE
  300 CONTINUE
      RETURN
      END
C
      SUBROUTINE REGRES
```

```
C
C*****  THIS SUBROUTINE CALCULATES THE REGRESSION COEFFICIENTS OF THE
C*****  RESPONSE SURFACES
C
      DIMENSION XBAR(30),STD(30),RX(900),RT(465),BV(30),DV(30),TV(30),
     &          ISAVE(30),RY(30),SB(30),ANS(10),XE(1000),YE(1000)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BL6/B(10,30)
      COMMON/BL9/EX(1000),EY(1000)
C
      NXE=MEXP*N2D
      N2DP1=N2D+1
C
      WRITE (12,1450)
  500 DO 600 I=1,MEXP
      KX1=(I-1)*N2D+1
      KX2=I*N2D
      KY1=(I-1)*NR+1
      KY2=I*NR
      WRITE (12,1600) I,(EX(J),J=KX1,KX2),(EY(J),J=KY1,KY2)
  600 CONTINUE
C
      CALL  GMTRA(EX,XE,N2D,MEXP)
      CALL  GMTRA(EY,YE,NR,MEXP)
C
      DO 50 J=1,N2DP1
      ISAVE(J)=J
   50 CONTINUE
C
C
      WRITE (12,1150)
      DO 300 J=1,NR
      KXE=NXE
      KYE=MEXP*(J-1)
      DO 100 I=1,MEXP
      KXE=KXE+1
      KYE=KYE+1
      XE(KXE)=YE(KYE)
  100 CONTINUE
C
      CALL  CORRE(MEXP,N2DP1,1,XE,XBAR,STD,RX,RT,BV,DV,TV)
      CALL  ORDER(N2DP1,RT,N2DP1,N2D,ISAVE,RX,RY)
      CALL  MINV(RX,N2D,DET,DV,TV)
      CALL  MULTR(MEXP,N2D,XBAR,STD,BV,RX,RY,ISAVE,DV,SB,TV,ANS)
      WRITE (12,1160)(DV(L),L=1,N2D),ANS(1),J
      FVALU=ANS(10)
      WRITE (12,1170) FVALU
C
      DO 200 JCOEF=1,N2D
      B(J,JCOEF)=DV(JCOEF)
  200 CONTINUE
      B(J,N2DP1)=ANS(1)
  300 CONTINUE
C
 1450 FORMAT (//45X,'*****   REGRESSION TABLEAU   *****'//12X,'X1',12X,
     &'X2',11X,'X1SQ',10X,'X2SQ',10X,'X1X2',11X,'Y1',12X,'Y2',12X,'Y3',
     &12X,'Y4'/12X,'Y5',12X,'Y6')
 1600 FORMAT(/1X,I3,3X,9(F12.6,2X)/7X,9(F12.6,2X)/)
 1150 FORMAT (////1X,'THE ESTIMATED RESPONSE SURFACES ARE :')
```

```
 1160 FORMAT (/5X,F12.6,' X(1) + ',F12.6,' X(2) + ',F12.6,
     &' X(1)SQ + ',F12.6,' X(2)SQ + ',F12.6,' X(1)*X(2) + ',
     &F12.6,' = Y(',I1,')')
 1170 FORMAT (//5X,'F VALUE  =  ',F12.6)
      RETURN
      END
C
      SUBROUTINE FSBPT1(XT,Y,IOUT)
C
C*****  THIS SUBROUTINE INDICATES IF THE CURRENT POINT IS FEASIBLE
C*****           IOUT=1   :  VIOLATES THE IMPLICIT CONSTRAINT(S)
C*****           IOUT=2   :  VIOLATES THE EXPLICIT CONSTRAINT(S)
C
      DIMENSION XT(30),Y(10),YY(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BL15/KDUPK,DUPK
      IOUT=2
      DO 100 J=1,N
      IF (XT(J).LT.XLP(J)) GO TO 500
      IF (XT(J).GT.XUP(J)) GO TO 500
  100 CONTINUE
      IOUT=1
C
      I=1
      IF (KDUPK.NE.1) WRITE (12,1120)  I
      CALL SIMUL(XT,Y,NR,1)
      REWIND 10
      IF (KDUPK.EQ.1) GO TO 190
      DO 160 I=2,KDUPK
      WRITE (12,1120) I
      CALL SIMUL(XT,YY,NR,2)
      REWIND 10
      DO 140 J=1,NR
      Y(J)=Y(J)+YY(J)
  140 CONTINUE
  160 CONTINUE
      DO 180 J=1,NR
      Y(J)=Y(J)/DUPK
  180 CONTINUE
C
  190 DO 200 J=2,NR
      IF (Y(J).LT.YL) GO TO 500
      IF (Y(J).GT.YU) GO TO 500
  200 CONTINUE
      IOUT=0
C
 1120 FORMAT (/1X,'** SIMUL.',I3,' **'/)
  500 RETURN
      END
C
      SUBROUTINE FSBPT2(XT,Y,IOUT)
C
C*****  THIS SUBROUTINE CALCULATES THE RESPONSES FROM THE REGRESSION
C*****  FUNCTIONS AND INDICATES IF THE CURRENT POINT IS FEASIBLE
C*****           IOUT=1   :  VIOLATES THE IMPLICIT CONSTRAINT(S)
C*****           IOUT=2   :  VIOLATES THE EXPLICIT CONSTRAINT(S)
C
```

```
      DIMENSION XT(30),Y(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BL6/B(10,30)
      IOUT=2
C
      DO 100 J=1,N
      IF (XT(J).LT.XLP(J)) GO TO 500
      IF (XT(J).GT.XUP(J)) GO TO 500
  100 CONTINUE
      IOUT=1
C
      DO 300 I=2,NR
      Y(I)=B(I,N2DP1)
      DO 200 J=1,N2D
      Y(I)=Y(I)+B(I,J)*XT(J)
  200 CONTINUE
      IF (Y(I).LT.YL) GO TO 500
      IF (Y(I).GT.YU) GO TO 500
  300 CONTINUE
      Y(1)=B(1,N2DP1)
      DO 350 J=1,N2D
      Y(1)=Y(1)+B(1,J)*XT(J)
  350 CONTINUE
      IOUT=0
C
  500 RETURN
      END
C
      SUBROUTINE WORST(YT,MIN)
C
C*****  THIS SUBROUTINE FINDS OUT THE WORST POINT IN THE
C*****  POLYGON
C
      DIMENSION YT(30)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
C
      MIN=1
      YMIN=YT(1)
      DO 50 J=2,NP
      IF (YT(J).GE.YMIN) GO TO 50
      MIN=J
      YMIN=YT(J)
   50 CONTINUE
      RETURN
      END
C
      SUBROUTINE DATA(NV,DV)
C
C*****  THIS SUBROUTINE IS CALLED BY THE SSP SUBROUTINE 'CORRE'
C*****  TO READ IN THE EXPERIMENT DATA NEEDED FOR REGRESSION
C
      DIMENSION DV(1)
    1 FORMAT (12F6.0)
      READ (3,1) (DV(J),J=1,NV)
      WRITE (1) (DV(J),J=1,NV)
      RETURN
      END
```

```
C
      SUBROUTINE WRITE(XW,YW,N,NR,IOUT)
      DIMENSION XW(30),YW(10)
C
      IF (IOUT.NE.0) GO TO 500
C
      DO 100 J=1,N
      WRITE (12,1100) J,XW(J)
  100 CONTINUE
      DO 200 J=1,NR
      WRITE (12,1200) J,YW(J)
  200 CONTINUE
      GO TO 1000
C
  500 DO 600 J=1,N
      WRITE (12,1600) J,XW(J)
  600 CONTINUE
      DO 700 J=1,NR
      WRITE (12,1700) J,YW(J)
  700 CONTINUE
C
 1100 FORMAT (30X,'X(',I1,') = ',F14.6)
 1200 FORMAT (30X,'Y(',I1,') = ',F14.6)
 1600 FORMAT (80X,'X(',I1,') = ',F14.6)
 1700 FORMAT (80X,'Y(',I1,') = ',F14.6)
 1000 RETURN
      END
C
      SUBROUTINE RECORD(XP,Y,KX,KY)
      DIMENSION XP(30),Y(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BL9/EX(1000),EY(1000)
C
      DO 100 J=1,N2D
      KX=KX+1
      EX(KX)=XP(J)
  100 CONTINUE
      DO 200 J=1,NR
      KY=KY+1
      EY(KY)=Y(J)
  200 CONTINUE
      RETURN
      END
C
      SUBROUTINE REGION(XP)
      DIMENSION XP(10),JREGN(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BL11/JSECT(10),DAP(10)
      COMMON/BL12/KEXRG(1000)
      COMMON/BL13/IREGN,MILXR
C
      DO 300 J=1,N
      KREGN=(XP(J)-XLP(J))/DAP(J)
      JREGN(J)=KREGN+1
      IF (KREGN.EQ.JSECT(J)) JREGN(J)=JSECT(J)
  300 CONTINUE
C
```

```
      K=JREGN(N)
      DO 500 JP=1,NM1
      J=N-JP
      K=(K-1)*JSECT(J)+JREGN(J)
  500 CONTINUE
C
      KEXRG(K)=KEXRG(K)+1
      RETURN
      END
C
      SUBROUTINE COMPEN(KX,KY,ITRY,XLR,XUR,Y1MAX)
      DIMENSION XP(10),Y(10),XLR(10),XUR(10),XLRG(10),XURG(10)
      COMMON N,AN,NM1,N2D,N2DP1,NR,M,MEXP,NP,ANP,NPM1,ANPM1
      COMMON/BLEE1/XLP(10),XUP(10)
      COMMON/BLEE3/XL(10),XU(10),YL,YU
      COMMON/BL11/JSECT(10),DAP(10)
      COMMON/BL12/KEXRG(1000)
      COMMON/BL13/IREGN,MIEXR
C
      ITRY=0
      ICOMP=0
      DO 500 K=1,IREGN
      IF(KEXRG(K).GE.MIEXR) GO TO 500
      KA=K
      DO 300 J=1,N
      KB=MOD(KA,JSECT(J))
      IF(KB.EQ.0) KB=JSECT(J)
      KA=(KA-KB)/JSECT(J)+1
      AKB=KB
      XLRG(J)=XLP(J)+DAP(J)*(AKB-1.)
      XURG(J)=XLP(J)+DAP(J)*AKB
      XP(J)=(XLRG(J)+XURG(J))/2.
  300 CONTINUE
C
      MEXP=MEXP+1
      ICOMP=ICOMP+1
      WRITE (12,1400) ICOMP
      CALL SECIZE(XP)
      CALL FSBPTI(XP,Y,IOUT)
      CALL RECORD(XP,Y,KX,KY)
      CALL WRITE(XP,Y,N,NR,ICUT)
      IF (IOUT.EQ.1) GO TO 500
      IF (Y(1).LE.Y1MAX) GO TO 500
      Y1MAX=Y(1)
      DO 400 J=1,N
      XLR(J)=XLRG(J)
      XUR(J)=XURG(J)
  400 CONTINUE
      ITRY=1
  500 CONTINUE
C
 1400 FORMAT(/1X,I4,' TH COMPENSATE POINT'/)
      RETURN
      END
C
      SUBROUTINE SIMUL(XS,Y,IIP1,KTER)
      DIMENSION XS(10),Y(20)
      COMMON /SUMMRY/ TKACT,TKDET,TITSNK,TITCUT,TNEUAT,TNESUC,TNUEAL,
     1    TLOST,TWAST,RUN
```

DISTRIBUTION LIST

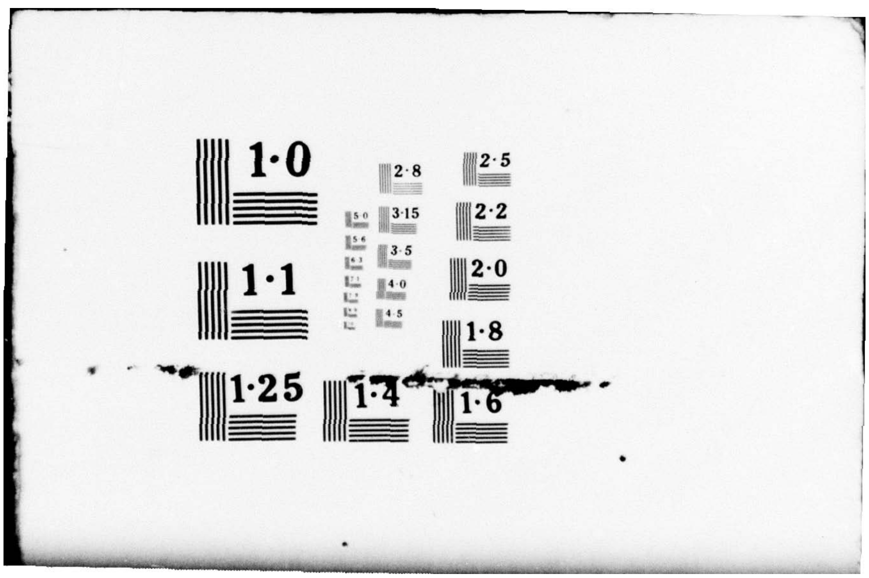
Name	Number of copies
Office of Naval Research Arlington, VA 22217	
Code 431	2
Code 434	1
Code 436	1
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
ONR Branch Office 536 South Clark Street Chicago, Illinois 60605	1
Naval Research Laboratory Code 2627 Washington, DC 20375	6
ONR Resident Representative Purdue University Room 85 - Graduate House West Lafayette, Indiana 47907	1
Desmatics, Inc. PO Box 618 State College, PA 16801	1
George S. Fishman Curriculum in Operations Research and Systems Analysis Phillips Annex University of North Carolina at Chapel Hill Chapel Hill, NC 27514	1
Donald P. Gaver Naval Postgraduate School Operations Research Department Monterey, CA 93940	1
Toke Jayachandran Naval Postgraduate School Monterey, CA 93940	1
D. L. Iglehart Department of Operations Research Stanford University Stanford, CA 94305	1

AD-A062 389   NOTRE DAME UNIV IND DEPT OF AEROSPACE AND MECHANICA--ETC F/G 12/2
OPTIMIZATION OF MULTIPLE-RESPONSE SIMULATION MODELS.(U)
OCT 78 W E BILES   N00014-76-C-1021

UNCLASSIFIED   NL

2 OF 2
ADA
062389

END
DATE
FILMED
3 -79
DDC


1·0
2·8
2·5
5·0
3·15
2·2
5·6
3·5
6·3
2·0
4·0
1·1
4·5
1·8
1·25
1·4
1·6

DISTRIBUTION LIST

Name	Number of copies
Dr. Averill M. Law 1513 University Avenue Department of Industrial Engineering University of Wisconsin Madison, Wisconsin 53706	1
Dr. Peter A. W. Lewis Naval Postgraduate School Department of Administrative Sciences Monterey, CA 93940	1
Dr. Richard E. Nance Computer Science, Head Virginia Polytechnic Institute and State University Blacksburg, VA 24061	1
Dr. A. Alan B. Pritsker Department of Industrial Engineering Purdue University West Lafayette, Indiana 47907	1
J. William Schmidt Department of Industrial Engineering and Operations Research Virginia Polytechnic Institute and State University Blacksburg, VA 24061	1
Naval Surface Weapons Center Code K-30 (Kulp and Patton, and Miller) Dahlgren, VA 22448	3
David W. Taylor Naval Ship Research & Development Center Code 1806 (S. Goldstein) Bethesda, MD 20084	1
Naval Research Laboratory Code 5308 Washington, DC 20375	1
Naval Ocean Systems Center Code 16 San Diego, CA 92152	1